Universidad Carlos III de Madrid

Escuela Politécnica Superior

Grado en Ingeniería Informática

Proyecto Fin de Grado

Sistema de recolección de objetos mediante visión artificial y planificación automática

Autor: Marcos Zimmermann Casado

Tutor: Moisés Martínez Muñoz

Resumen

Este trabajo consiste en el desarrollo de un sistema de control para un robot humanoide mediante la utilización de planificación automática y visión artificial. Este sistema debe permitir al robot recoger uno o varios objetos presentes en el entorno y transportarlos a otro lugar previamente seleccionado. Para ello el robot utilizará la planificación automática y la visión artificial con el fin de resolver esta tarea de forma automática. Además el sistema podrá ser configurado a priori por un usuario humano, indicando al robot las características de los objetos que debe recoger y transportar. Para comprobar el correcto funcionamiento del sistema se ha utilizado el robot humanoide NAO, el cual ha tenido que resolver un conjunto de problemas en entornos reales de forma autónoma.

Abstract

This work consists in the development of a control system for a humanoid robot using automated planning and computer vision. This system must allow to pick one or various objects from the environment and carry them on to another place selected previously. According to this, the robot will use automated planning and computer vision to solve this task. Besides, the system could be configured a priori by a human user, providing to the robot information about the characteristics of the objects that must pick and carry on. Finally, an Humanoid Robot has been used to test the system performance, which has solved a set of problems in a real time environment autonomously.

Índice General

Resumen **1**

Abstract **2**

Capítulo 1: Introducción **10**

1.1 Descripción del problema **10**

1.2 Motivación **11**

1.3 Objetivos del trabajo **12**

1.4 Estructura del documento **13**

Capítulo 2: Estado del Arte **14**

2.1 Arquitecturas de control **14**

2.2.1 Arquitecturas deliberativas 14

2.2.1 Arquitecturas reactivas 15

2.2.1 Arquitecturas híbridas 16

2.2.1 Arquitecturas basadas en comportamientos 17

2.2 Visón artificial **17**

2.2.1 Reconocimiento 18

2.2.2 Modelos de color 19

2.3 Planificación automática **21**

2.3.1 PDDL 22

2.3.2 Metric-FF 24

2.4 PELEA **24**

2.4.1 Ciclo de ejecución de PELEA 26

2.5 Robots **27**

2.5.1 Robots móviles 29

2.5.2 Robots humanoides 29

2.5.3 Robot NAO 31

Capítulo 3: Descripción del sistema **33**

3.1 Introducción **33**

3.2 Análisis del sistema **34**

3.2.1 Descripción de las características funcionales 34
3.2.2 Restricciones del sistema 35
3.2.3 Entorno operacional 37
3.2.4 Especificación de casos de uso 39
3.2.5 Especificación de requisitos 55
3.3 Diseño del sistema 66
3.3.1 Arquitectura del sistema 66
3.3.2 Descripción de componentes 67
3.3.3 Descripción general del funcionamiento del sistema 93
Capítulo 4: Experimentación 96
4.1 Entorno de pruebas 96
4.2 Pruebas del simulador 98
4.3 Experimentos 100
4.2.1 Experimento 1 100
4.2.1 Experimento 2 104
Capítulo 5: Gestión del proyecto 107
5.1 Descripción de las fases del proyecto 107
5.2 Planificación 108
5.3 Presupuesto 109
Capítulo 6: Conclusiones y trabajos futuros 111
6.1 Conclusiones generales 111
6.2 Conclusiones referentes a los objetivos 112
6.3 Trabajos futuros 114
Capítulo 7: Anexos 116
7.1 Manual de instalación 116
7.2 Manual de usuario 116
7.3 Librerías del NAOqi usadas 117
7.4 Problemas usados en la experimentación 118
7.4.1 Problema experimento 1: 118
7.4.2 Problema experimento 2: 120
Glosario 122

Acrónimos 123

Bibliografía 125

Índice de figuras

Figura 1 – Robot cogiendo el objeto sobre la caja. 11

Figura 2 – Robot Shakey (a) Esquema de la arquitectura deliberativa (b) 15

Figura 3 – Arquitectura reactiva. 15

Figura 4 – Arquitectura híbrida. 16

Figura 5 – Arquitectura basada en comportamientos. 17

Figura 6 – Fases reconocimiento en imágenes. 18

Figura 7 – Representación geométrica espacio RGB (a) Rango de colores RGB (b) 20

Figura 8 – Representación geométrica espacio HSV 21

Figura 9 – Acción de STRIPS 23

Figura 10 – Acción de PDDL 23

Figura 11 – Arquitectura de pelea 25

Figura 12 – Ciclo de ejecución pelea1level 26

Figura 13 – Robots industriales manipuladores en una cadena de montaje (a) MQ-9 Reaper armado en vuelo (b) 27

Figura 14 – Robot de Servicio Roomba (a) Robot educacional Play-I (b) 28

Figura 15 – Robot modular Atron (a) Robot colaborativo FRIDA (b) 28

Figura 16 – Robot Asimo bajando escaleras (a) Robot NAO ayudando a un niño con desórdenes sociales (b) 30

Figura 17 – AR robot realizando tareas domésticas (a) Actroid-DER 01(b) 31

Figura 18 – Hardware del robot NAO 32

Figura 19 – Diagrama alto nivel del sistema 34

Figura 20 – Rango de las cámaras del NAO 36

Figura 21 – Diagrama de casos de uso del sistema 40

Figura 22 – Subsistemas 66

Figura 23 – Diagrama de componentes del sistema 67

Figura 24 – Botones táctiles del robot situados en la cabeza 69

Figura 25 – Cámara del robot en modo vídeo con área de captura 1 (a) Cámara del robot en modo vídeo con área de captura 2(b) 69

Figura 26 – Diagrama de flujo de la tarea de selección de objetos 71

Figura 27 – Diagrama de flujo de la tarea de reconocer objeto 72

Figura 28 – Objeto reconocido 73

Figura 29 – Imagen capturada por una de las cámaras del robot NAO 73

Figura 30 – Diagrama de flujo del procesamiento de una imagen 74

Figura 31 – Imagen original (a) Imagen de OpenCV (b) 75

Figura 32 – Imagen OpenCV cambiada a BGR (a) Imagen difuminada (b) 76

Figura 33 – Imagen en formato HSV (a) Imagen en blanco y negro (b) 76

Figura 34 – Imagen con suavizado, dilatado y erosionado (a) Centroide encontrado sobre la imagen original (b) 76

Figura 35 – Robot Nao sosteniendo un pincel envuelto 77

Figura 36 – Objeto guardado mediante visón 78

Figura 37 – Contenedor reconocido 79

Figura 38 – Cámaras del robot (a) Límite para cambiar de cámara del robot(b) 80

Figura 39 – Robot usando cámara 2 con cabeza inclinada (a) Robot usando cámara 2 con cabeza en posición normal (b) 81

Figura 40 – NAO intentando coger el objeto (a) Robot Nao sosteniendo el objeto (b) 82

Figura 41 – NAO intentando depositar el objeto en el contenedor (a) Robot Nao ha depositado el objeto en el contenedor (b) 83

Figura 42 – Diferencia de posición del objeto respecto al robot 84

Figura 43 – Imagen seccionada en 8 áreas 85

Figura 44 – Posición antes del giro (a) Giro de 90º fallido (b) Corrección del giro por los valores del giroscopio(c) 87

Figura 45 – Objetos y predicados del dominio de planificación automática 88

Figura 46 – Acción "move" en el dominio de planificación automática 89

Figura 47 – Acción "turn" en el dominio de planificación automática 90

Figura 48 – Acción "catch" en el dominio de planificación automática 91

Figura 49 – Acción "drop" en el dominio de planificación automática 92

Figura 50 – Funcionamiento del sistema 95

Figura 51 – Entorno de pruebas 96

FIgura 52 – Caja – soporte del objeto (a) Caja con el objeto colocado (b) 98

Figura 53 – Caja - contenedor del objeto 98

Figura 54 – Imagen original (a) Imagen de OpenCV (b) 99

Figura 55 – Imagen OpenCV cambiada a BGR (a) Imagen difuminada (b) 99

Figura 56 – Imagen en formato HSV (a) Imagen en blanco y negro (b) 99

Figura 57 – Imagen con suavizado, dilatado y erosionado (a) Centroide encontrado sobre la imagen original (b) 100

Figura 58 – Esquema del problema (a) Imagen real del problema (b) 101

Figura 59 – Esquema del problema 2 (a) Imagen real del problema 2 (b) 104

Figura 60 – Modelo de ciclo de vida en espiral 108

Figura 61 – Planificación de tiempo para el proyecto (Diagrama de Gantt) 109

Figura 62 – Rutas de los ficheros de PELEA 116

Figura 63 – Problema 1 experimentación – parte 1 118

Figura 64 – Problema 1 experimentación – parte 2 119

Figura 65 – Problema 2 experimentación – parte 1 120

Figura 66 – Problema 2 experimentación – parte 2 121

Índice de tablas

Tabla 1 – Matriz de trazabilidad Casos de uso - Requisitos 65

Tabla 2 – Resultados experimento 1..... 102

Tabla 3 – Resultados experimento 2..... 105

Tabla 4 - Costes estimados de los recursos físicos 109

Tabla 5 - Costes estimados de los recursos físicos 110

Tabla 6 - Costes estimados del proyecto 110

Capítulo 1: Introducción

Los avances tecnológicos de los últimos años han mejorado la capacidad que tienen los sistemas informáticos para poder extraer y analizar la información del mundo real, utilizándola para resolver problemas de forma automática, de forma similar a como los resolvería el ser humano. Un claro ejemplo de estos avances es MAPGEN[1], una aplicación que permite controlar un robot en la superficie de otro planeta mediante inteligencia artificial o BEAR[2], un sistema que permite rescatar personas heridas que se encuentren en situaciones de riesgo.

Existen diferentes formas de implementar un sistema de control para poder conseguir que un robot se comporte de manera autónoma. Dependiendo de la manera en la cual los sistemas de control toman las decisiones, pueden ser clasificados en: reactivos, deliberativos, híbridos o basados en comportamientos. Teniendo en cuenta las diferentes características de cada uno de ellos, el que mejor se adapta a los objetivos de este trabajo es el sistema híbrido. Este combina las características de los sistemas reactivos y deliberativos. Es decir, está compuesto por una capa de alto nivel o capa deliberativa que se encarga de tomar las decisiones de alto nivel, aquella s que están relacionadas con los modelos de razonamientos humanos; y otra capa de bajo nivel o nivel reactivo que se encarga de adaptar los comportamientos generados por la capa deliberativa adaptándose a las necesidades del entorno, así como gestionando los diferentes sensores y actuadores de los cuales estará provisto el robot.

1.1 Descripción del problema

En este trabajo se pretende resolver un problema de recogida de objetos en un entorno variable, mediante el cual un robot humanoide sea capaz de recoger un conjunto de objetos del entorno y depositarlos en otro lugar del mismo. El robot debe ser capaz de localizar los diferentes objetos del entorno teniendo en cuenta una serie de características, como por ejemplo el color o la forma. Debido a la complejidad del problema se van a aplicar una serie de simplificaciones mediante las cuales el problema será más sencillo, desde el punto de vista computacional. A continuación se realiza una descripción de las diferentes simplificaciones que se han llevado a cabo:

- El entorno (habitación, pasillo, laboratorio, etc) será representado mediante un conjunto de de casillas uniformes entre las cuales podrá moverse el robot.

- El estado inicial del entorno y de los diferentes objetos son conocidos a priori, es decir, el robot tiene un conocimiento completo del estado del mundo. Además el entorno sólo sufrirá modificaciones cuando sean realizadas por el robot, aunque se podría introducir un grado de fiabilidad en la información que se conoce a priori, como por ejemplo el color o la forma de los objetos, pudiendo el robot detectar si esta información es o no correcta.
- Los objetos estarán todos posicionados a cierta altura de forma que el robot humanoide no tenga que agacharse o realizar movimientos muy complicados que aumenten más el grado de dificultad del problema. En la Figura 1 se presenta la estructura en la cual se encontrarán los objetos de forma que el robot pueda cogerlos sin necesidad de agacharse.

Figura 1 – Robot cogiendo el objeto sobre la caja.

Este problema ofrece ciertas posibilidades para construir situaciones muy diferentes, como aumentar el tamaño del entorno, su estructura, así como el número o características de los objetos que deben ser recogidos.

1.2 Motivación

Conseguir que un robot humanoide sea capaz de realizar tareas complejas similares a las que puede realizar un humano es realmente interesante. Tareas que para nosotros los humanos, parecen triviales, tales como recoger un objeto, es un proceso realmente complejo para un sistema informático. Por ejemplo, para nosotros los humanos, existen numerosas tareas que nos resultan sencillas pero que llevan muchas otras tareas complejas por debajo que han sido perfeccionadas mediante la experiencia o el aprendizaje. Por ejemplo, para recoger un objeto, ya tenemos un conocimiento previo de cómo agarrarlo, la percepción de profundidad para ubicar la

mano en la posición adecuada, qué objeto es, qué forma tiene, qué color… Todas estas tareas que las realizamos de manera inconsciente para recoger un objeto, hacen falta ser modeladas y adaptadas para un sistema informático; éste proceso es realmente complejo, ya que es necesario convertir la información del mundo real que, a priori, un sistema informático no sabe interpretar, en valores que sí que sepa interpretar y procesar.

La motivación que me ha llevado a realizar este trabajo viene dada por los resultados del trabajo, que es conseguir que un robot, de manera autónoma, sea capaz de identificar un objeto, recogerlo y llevarlo a una posición establecida a priori. Se trataba de un proyecto muy llamativo a la par que complejo. Adicionalmente quería embarcarme en el mundo de la robótica y aprender cómo funciona un robot, aprender nuevos conceptos de la informática para mí, tales como la visión artificial, tratamiento de sensores, integración de software, interacción con la API de un robot, conocer las dificultades de procesar y adaptar la información del mundo real… Este trabajo era idóneo para ello, me iba permitir aprender sobre un gran variedad de campos complementarios a la formación académica recibida.

1.3 Objetivos del trabajo

El objetivo principal de este trabajo consiste en el desarrollo de un sistema de control para el robot NAO, mediante el cual pueda recoger una serie de objetos distribuidos por el entorno y depositarlos en otro lugar del entorno. Debido a la complejidad del objetivo principal, este va a ser dividido en pequeños objetivos de forma que sea más sencillo explicar la complejidad del trabajo:

- Un estudio previo de las técnicas y tecnologías a utilizar para conseguir los apartados expuestos a continuación.
- Desarrollo del sistema de movimiento. El robot debe ser capaz de conocer su ubicación (auto-localización) en el entorno de forma aproximada. Así como ser capaz de corregir los errores producidos por sus movimientos de forma automática, mediante la utilización del giroscopio y/o acelerómetro.
- Desarrollo del sistema de visión. Este debe ser capaz de procesar imágenes para identificar objetos.
- Desarrollo del sistema de recolección de objetos mediante la utilización de los brazos del robot y el sistema de visión del punto anterior.
- Integración del sistema de control con la arquitectura PELEA.

- Combinación de todos los sistemas anteriores para crear el sistema de control híbrido.
- Evaluación del sistema mediante experimentos para probar su correcto funcionamiento.

1.4 Estructura del documento

Este documento se divide en 7 capítulos. De los cuales el último alberga los distintos anexos del trabajo. A continuación se realiza una descripción detalla del contenido de cada uno de ellos:

1. El primer capítulo del documento presenta una breve introducción del trabajo, una descripción del problema que se intenta resolver con este trabajo, así como las distintas motivaciones que me han llevado a su realización y los distintos objetivos que fueron planteados al inicio del mismo.
2. El segundo capítulo del documento explica el Estado del Arte, en el que se plantea el estado de la cuestión del trabajo junto con las técnicas y tecnologías utilizadas.
3. El tercer capítulo del documento presenta una descripción detallada del sistema que ha sido desarrollado. Describiéndose sus restricciones, el entorno operacional, la especificación de casos de uso y los requisitos. Finalmente se describirá la arquitectura con sus componentes junto con el funcionamiento del sistema.
4. El cuarto capítulo del documento presenta la experimentación realizada sobre el sistema desarrollado en este trabajo. Se realiza una descripción detallada del entorno utilizado, otras pruebas realizadas durante el desarrollo del trabajo y los experimentos realizados junto con sus conclusiones. Para cada experimento se facilitará un video mostrando el funcionamiento del sistema.
5. El quinto capítulo del documento detalla la gestión del trabajo, con las fases que han tenido lugar desde su inicio hasta su final, toda la planificación llevada a cabo y el presupuesto necesario para llevar a cabo este trabajo.
6. El sexto capítulo del documento recoge las conclusiones obtenidas tras la realización del trabajo. En primer lugar se presentan las conclusiones generales, a continuación las conclusiones obtenidas para cada uno de los objetivos. Para finalizar el capítulo se exponen posibles trabajos futuros del trabajo.
7. El séptimo capítulo del documento incluye los anexos del trabajo, incluyendo un manual de instalación, un manual de usuario, los módulos del robot usados y los ficheros de los experimentos realizados. Al final del documento también se presentan el glosario, acrónimos y la bibliografía que ha sido usada en este trabajo.

Capítulo 2: Estado del Arte

En este capítulo se presenta el estado del arte relacionado con este trabajo. En primer lugar se presenta una introducción de las diferentes arquitecturas de control, seguido de una descripción detallada de algunas técnicas de visión artificial. A continuación, una descripción de la planificación automática. Después se describe la arquitectura PELEA, de qué se trata, para qué se puede utilizar y su funcionamiento. Por último, se realiza un análisis de los robots, los tipos existentes, hasta llegar a la clase de robot que se ha usado en este trabajo, el robot NAO.

2.1 Arquitecturas de control

A lo largo de los últimos años se han desarrollado diferentes paradigmas de control con el fin de conseguir que una máquina o un robot sea capaz de actuar de manera autónoma, dependiendo de cómo se toman las decisiones, las arquitecturas pueden ser deliberativas, reactivas, híbridas y basadas en comportamientos [3][4].

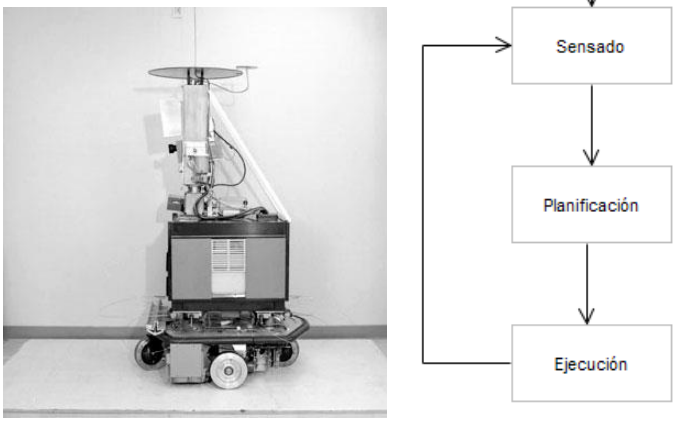
2.2.1 Arquitecturas deliberativas

Las arquitecturas deliberativas fueron desarrolladas a finales de 1960 siendo el robot Shakey[5], desarrollado por la universidad de Standford (Figura 2(a)), el primer sistema en utilizar una arquitectura de este tipo. La arquitectura está compuesta por tres fases: Sensado, Planificación y Ejecución. El sensado traduce la información del mundo real en una representación simbólica del mismo. El planificador toma la representación simbólica, junto con un objetivo y lo resuelve, obteniendo un plan capaz de resolver el problema propuesto. La ejecución se encarga de enviar al robot las acciones del plan capaz de resolver el problema. El funcionamiento de la arquitectura se presenta en el diagrama presentado en la Figura 2(b). Esta arquitectura se basa en construir un modelo del mundo de forma detallada y planificar de forma cuidadosa qué acción realizar. Esta aproximación presenta los siguientes inconvenientes:

- La planificación consume un número de recursos muy elevado, y podría bloquear al robot esperando a que termine de planificar.
- La ejecución del plan sin volver a tener en cuenta el sensado podría poner en riesgo la integridad del robot si se trata de un entorno dinámico.

Debido a estos problemas, esta arquitectura no es adecuada para sistemas que tienen que interactuar con un entorno dinámico, ya que si cambia su entorno el plan generado podría dejar

de ser válido. Además, tampoco es adecuada para sistemas que tengan que reaccionar de manera rápida, pero sí que son efectivas para resolver problemas complejos.



(a) (b)

Figura 2 – Robot Shakey (a) Esquema de la arquitectura deliberativa (b)

2.2.1 Arquitecturas reactivas

En 1986 Rodney Brooks publicó un artículo donde presentaba el concepto de arquitectura reactiva[6]. Esta arquitectura se compone de un conjunto de capas, cada una de las cuales implementa una máquina de estados finita, que está conectada directamente con los sensores y actuadores del robot. Estas máquinas de estado definen reglas, que dependen enteramente de los valores de los sensores; si se satisface una de estas reglas, se llevará a cabo la acción relacionada. El esquema general de una arquitectura reactiva se presenta en la Figura 3.

Figura 3 – Arquitectura reactiva.

Esta arquitectura presenta los siguientes problemas:

- Difícil de realizar tareas en las que el objetivo final necesitase de muchas acciones.
- Las máquinas de estados pueden ser muy complejas de implementar dependiendo de la acción a realizar o de la estructura del sistema a controlar.

Debido a estas limitaciones esta arquitectura no es adecuada para la resolución de problemas difíciles desde el punto de vista computacional o donde el objetivo último requiera muchos pasos intermedios. Sin embargo, si es adecuada para entornos dinámicos, ya que está continuamente recibiendo los valores de los sensores y actuando de acuerdo a ellos; también es adecuado para situaciones en las que hay que tomar decisiones de forma rápida.

2.2.1 Arquitecturas híbridas

Debido a los problemas que suponían las arquitecturas deliberativas y reactivas, se optó por un acercamiento que combinase lo mejor de ambas, reacción rápida y planificación para encontrar una solución global al problema. Esta arquitectura se puede descomponer en cuatro fases: Sensado, Planificación, Control y Ejecución. El sensado percibe la información del mundo y la envía al control de la arquitectura. La planificación se encarga de las tareas complejas computacionales, así como resolver el problema, en algunas ocasiones se utilizan planificadores de corto alcance combinado con otros de largo alcance. La ejecución envía las acciones recibidas por el módulo de control que el robot las pueda realizar. Control se encarga de decidir gestionar y decidir qué acción tomar, si una reactiva o una deliberativa recibida por el planificador; la acción se le envía al módulo de ejecución. El esquema general de una arquitectura híbrida se presenta en la Figura 4.

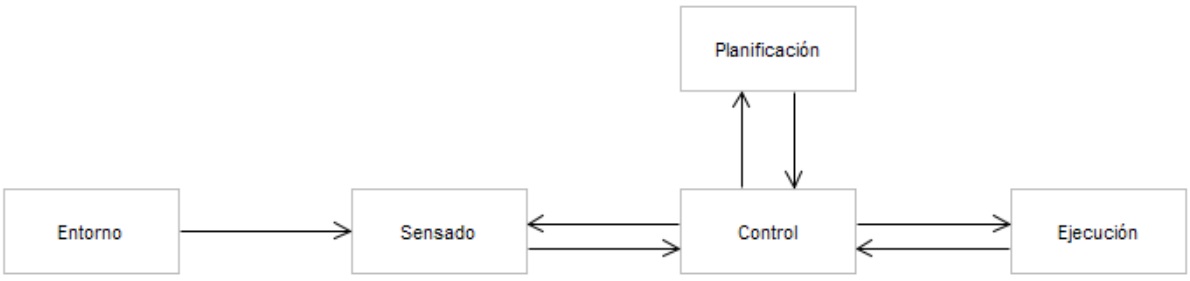


Figura 4 – Arquitectura híbrida.

Esta arquitectura resulta ser bastante efectiva, pero la parte compleja es el mecanismo de control, ya que es la que tiene que decidir qué acción (reactiva o deliberativa) deberá realizar el sistema en el momento adecuado.

2.2.1 Arquitecturas basadas en comportamientos

Esta arquitectura está compuesta por un conjunto de comportamientos. Cada comportamiento es un proceso independiente capaz de resolver ciertas metas. Por ejemplo evitar obstáculos, cuya meta es prevenir colisiones; caminar en línea recta, cuya meta es avanzar o coger un objeto, cuya meta es tener el objeto cogido. Dentro de los comportamientos, algunos tomarán exclusivamente de los sensores (primera capa) y enviarán su señal modificada a otros comportamientos o a los actuadores del robot, otros se comunicarán exclusivamente entre comportamientos (segunda capa) y por último otros recibirán la información exclusivamente de comportamientos y la enviarán a los actuadores del robot (tercera capa). Teniendo en cuenta esto, la arquitectura puede ser definida como una red de comportamientos que se comunican entre ellos. La Figura 5 presenta un ejemplo de una arquitectura compuesta por 6 comportamientos.

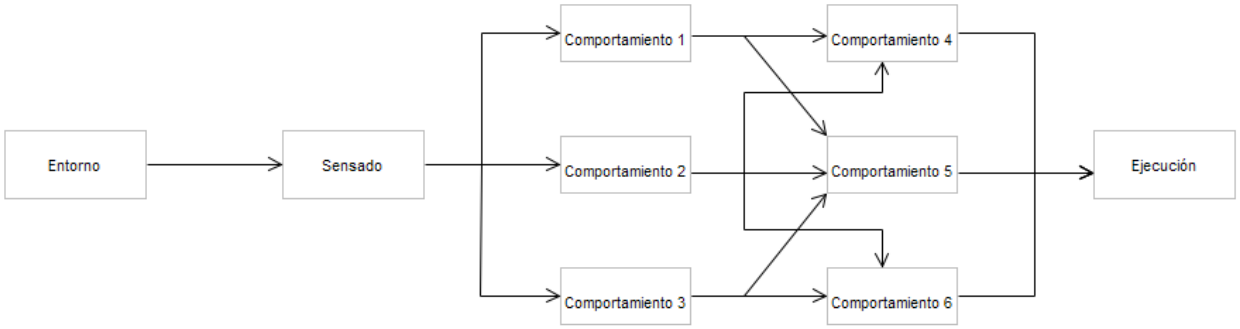


Figura 5 – Arquitectura basada en comportamientos.

Este tipo de arquitectura pretende tener un funcionamiento más uniforme que las arquitecturas híbridas, ofreciendo un sistema paralelizable, distribuido y activo, capaz de reaccionar a las demandas en tiempo real del sistema.

2.2 Visón artificial

La visión artificial es el área de la inteligencia artificial que intenta imitar el proceso de vision humano. Durante los últimos años se han desarrollado un conjunto de métodos que incluyen adquisición, procesado, analizado de imágenes u datos multidimensionales del mundo real, cuyo objetivo es adaptar la información del mundo real (generalmente imágenes) en valores numéricos

o información simbólica[7]. La visión artificial se puede aplicar en muchos campos y en muy variadas tareas. Algunas de las aplicaciones en las cuales ha sido aplicada la visión artificial son las siguientes:

- Control de procesos: por ejemplo mediante robots industriales[8].
- Navegación: por ejemplo mediante vehículos autónomos o robots móviles[9].
- Detección de eventos: por ejemplo mediante rastreo visual o contar personas[10].
- Organizar información: por ejemplo para indizar bases de datos de imágenes[11].
- Modelar objetos o entornos: por ejemplo análisis de imagen médica[12] o modelado topográfico.
- Interacción: por ejemplo la entrada de un dispositivo para la interacción entre hombre – máquina[13].
- Inspección automática: por ejemplo en cadenas de montaje[14].

2.2.1 Reconocimiento

Es la tarea más utilizada en el campo de la visión artificial, se trata del procesado de imagen cuyo objetivo es determinar si una imagen contiene un objeto, característica o actividad. Las técnicas actuales permiten reconocer objetos, colores, caras humanas, caracteres escritos a máquina o a mano, vehículos. Para poder reconocer los elementos anteriores, es necesario que se den unas condiciones adecuadas de iluminación, entorno y posición relativa a la cámara. Este tipo de reconocimiento, para los humanos es trivial pero se trata de una tarea compleja de resolver por los sistemas de visión artificial si se quiere realizar de manera genérica y en condiciones variables.

La estructura básica de un proceso de reconocimiento por visión artificial se presenta en la Figura 6.

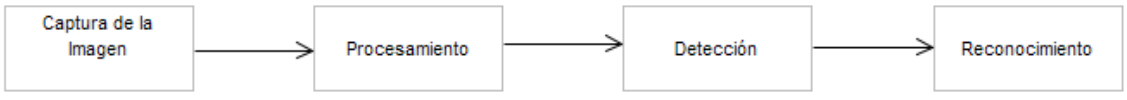


Figura 6 – Fases reconocimiento en imágenes.

- Captura de la imagen: Esta fase consiste en capturar la imagen en 2D o 3D, o secuencia de video donde se quiere analizar algún elemento.
- Procesamiento: Esta fase consiste en modificar la imagen original con el fin de facilitar las tareas de detección y reconocimiento. Dentro del procesamiento de imagen, estas son

algunas de las técnicas más usadas: Transformaciones de modelos de color, histograma de grises, suavizado, difuminado, agudizamiento (sharpening), filtrado de color, dilatación, erosionado, imagen en escala de grises…

- Detección: Esta fase consiste en identificar una o varias instancias de algún elemento conocido en la imagen. Por ejemplo, ejemplo en el caso de detección de objetos, el sistema deberá encontrar objetos de una clase conocida (vasos), pudiendo determinar su ubicación y en algunos casos sus límites. Algunas de las técnicas más utilizadas para la detección en imágenes son: detección de vértices mediante filtros Canny, variaciones entre píxeles (distancia, color, proximidad, grupos), entrenamiento por formas, búsqueda de contornos, etc.
- Reconocimiento: Esta fase consiste en identificar qué tipo de instancia se encuentra en la imagen. Por ejemplo, en el caso de reconocimiento de objetos, el sistema deberá decir qué tipo de instancias son los objetos. Siguiendo el ejemplo de detección, sería capaz de indicar que el objeto se trata de un vaso, una mesa, un plato, además de facilitar información de la posición del mismo. Algunas de las técnicas mas utilizadas para el reconocimiento en imágenes son: igualación de vértices, búsqueda dividir-y-conquistar, igualación de escala de grises, igualación de gradiente, búsqueda y comparación de bases de datos de modelos, árboles de interpretación, hipótesis y test, consistencia de poses, clustering de poses, hashing geométrico, SIFT, SURF, etc.

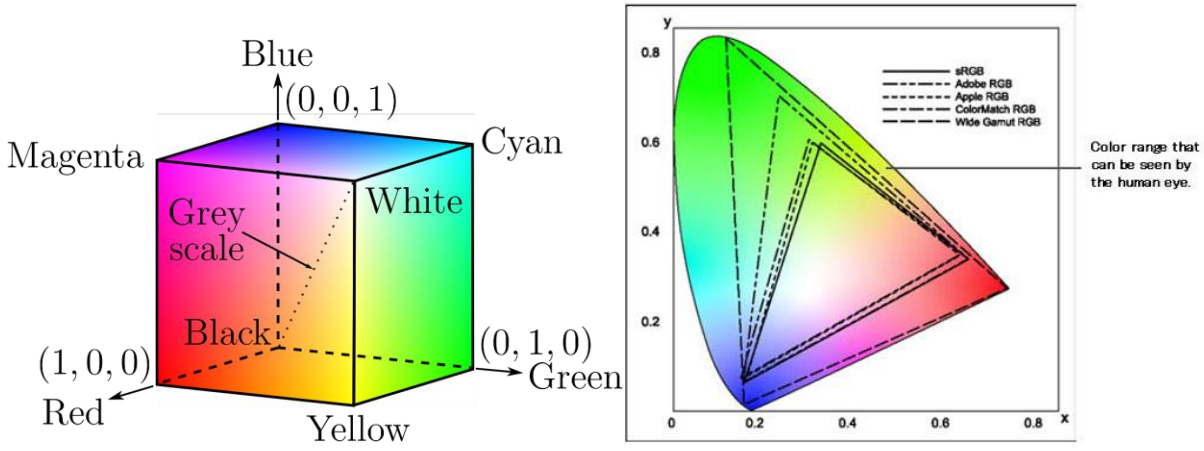
2.2.2 Modelos de color

La visión del color es la habilidad que presenta un organismo o máquina de distinguir objetos basándose en la longitud de onda de la luz que reflejan, emiten o transmiten. Actualmente, los sistemas de procesamiento de imagen son capaces de tratar las imágenes a color, pero antiguamente, por razones de rendimiento las imágenes se procesaban en blanco y negro; actualmente también se utiliza el procesamiento en blanco y negro para diferenciar colores o en sistemas embebidos que disponen de un procesamiento limitado. Los colores se pueden medir y cuantificar de diversas maneras; la percepción humana del color se trata de un proceso subjetivo donde el cerebro responde al estímulo que es producido cuando la luz entrante reacciona con los diferentes tipos de células foto receptoras dentro del ojo. Cada persona percibe el mismo objeto iluminado de diferentes maneras.

En la retina del ojo humano las células fotorreceptoras son preferentemente sensitivas a los colores azul, verde y rojo[15]. Por este motivo, en los humanos el color se percibe mediante la combinación de estos tres colores. El modelo de color RGB es el modelo de color más utilizado en los sistemas computacionales, pero no es el modelo de color que mejor represente los colores. Existen otros modelos de representación del color como HSV, YUV, Cielab.

2.2.2.1 Modelo RGB

El modelo RGB está basado en la percepción humana del color, mediante la combinación de los colores rojo, verde y azul (Red, Green, Blue). Su representación geométrica consiste en un cubo, como se puede observar en la Figura 7(a), cuyas esquinas representan los colores Azul (0,0,1), Cyan (0,1,1), Magenta (1,0,1), Blanco (1,1,1), Negro (0,0,0), Verde (0,1,0), Rojo (1,0,0) y Amarillo (1,1,0) . Es un modelo de color aditivo, ya que la suma de los colores primarios (rojo, verde y azul) da como resultado el color blanco, mientras que la ausencia de ellos, el color negro; se puede apreciar en la Figura 7(a) que el color negro corresponde con el punto [0,0,0] (ausencia de colores rojo, verde y azul) y el color blanco corresponde con el punto [1,1,1] (máximo valor de los colores rojo, verdey azul). Aunque esté basado en la percepción humana del color, se trata de un modelo poco intuitivo, ya que las diferencias entre colores adyacentes no se pueden percibir y es complicado saber el color que se está utilizando mediante este modelo. Debido a las características de su representación, el modelo RGB no puede representar todos los colores percibidos por el ojo humano, para ello se crearon variantes como el sRGB (Figura 7(b)) para aumentar el número de colores que podían ser representados mediante este modelo.



(a) (b)

Figura 7 – Representación geométrica espacio RGB (a) Rango de colores RGB (b)

2.2.2.2 Modelo HSV

El modelo HSV representa los colores mediante tres componentes, tono, saturación y valor (Hue, Saturation, Value). Su representación geométrica, se presenta en la Figura 8, consiste en un cilindro, donde los valores del tono corresponden con el ángulo respecto al eje central vertical, la saturación con la distancia desde el eje central vertical y la saturación con la altura (eje vertical). Esta representación permite especificar de forma más sencilla los colores, ya que el valor importante para determinar un color se trata del tono, cuyo rango es de 0-360º. La paleta completa de color (rojo o verde o azul) se puede dividir en secciones de 120º. De esta manera se puede percibir las diferencias de color entre colores adyacentes lo cual hace que sea uno de los modelos de color más usado en visión artificial. Como desventaja, los valores escalan mal con los valores de saturación y valor. Puede representar toda la gama de color percibida por el ojo humano.

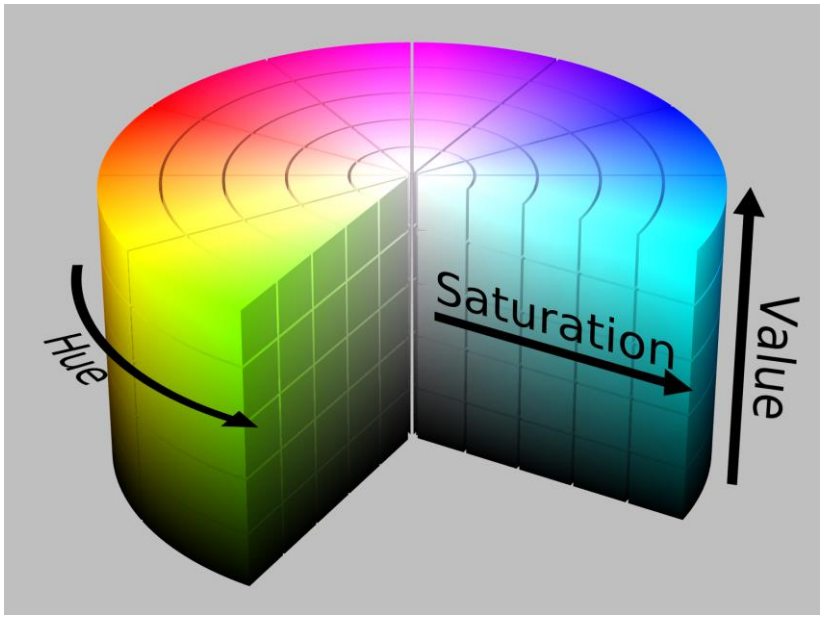


Figura 8 – Representación geométrica espacio HSV

2.3 Planificación automática

La planificación automática es una rama de la Inteligencia Artificial (IA), que trata de resolver problemas de tipo general que son representados mediante un lenguaje de alto nivel utilizando un algoritmo de tipo general. El primer sistema capaz de utilizar planificación automática como modelo de resolución de problemas fue el GPS[16]. Según *Artificial Immune Systems* (AIS) en el

informe de la 7ª Conferencia Internacional, ICARIS 2008, describe la planificación de la siguiente manera:

"[Planificación] es un área que estudia la generación automática de un plan para resolver un problema en un dominio específico. Básicamente, un plan es una secuencia de acciones proveídas por un planificador que, dado un estado inicial, intenta encontrar la forma de alcanzar unas condiciones meta. Los planificadores pueden ser dependientes de dominio o independientes de dominio. Los planificadores independientes de dominio no están atados a un dominio en concreto – pueden resolver problemas de diferentes dominios, dado un modelo del dominio en un lenguaje de entrada viable. [17]*."*

Desde el punto de vista teórico, la tarea de planificación puede ser definida como una tupla formada por 4 elementos: un conjunto de literales que son utilizados para representar el estado del mundo, un conjunto de acciones, un estado inicial y un conjunto de literales denominados metas. De manera más formal, la tarea de planificación π está definida por:

$$\pi = (L, O, I, G)$$

Donde:

- L: conjunto de literales (describen los estados)
- O: conjunto de operadores (acciones)
- I ⊆ L: estado inicial
- G ⊆ L: metas del problema

Teniendo en cuenta esta representación, un plan π que resuelve la tarea π, es una secuencia de acciones que ejecutadas secuencialmente transforman el estado inicial I en un estado meta donde todos los literales contenidos en G son ciertos.

2.3.1 PDDL

PDDL o Planning Domain Definition Languaje, es el lenguaje de representación utilizado por la comunidad de planificación para representar la tarea de planificación. En 1998 la AIPS Planning Competition Committee lanzó PDDL [18], basado en los lenguajes STRIPS [19] y ADL [20]. Desde entonces PDDL se ha convertido en un estándar para la presentación e intercambio de modelos de

dominios de planificación. STRIPS (Standford Research Institute Problem Solver) es un planificador automático desarrollado por Richard Fikes y Nils Nilsson en 1971[21]; el mismo nombre también hace referencia al lenguaje formal usado para los ficheros que requería el planificador. Este lenguaje se ha usado como base para otros lenguajes de planificación automática, como por ejemplo PDDL. Una ejemplo de una acción de STRIPS se puede ver en la Figura 9.

```
Actions:
        _turn(c1, o1, o2)_
        Preconditions: robot(c1, o1)
        Postconditions:  robot(c1 , o2), not robot(c1, o1)
)
```

Figura 9 – Acción de STRIPS

A continuación en la Figura 10 se muestra la misma acción en PDDL. Como se puede observar, la semántica es muy similar.

```
(:action turn
        :parameters (?c1 - cell ?o1 - orientation ?o2 - orientation)
        :precondition (and (robot ?c1 ?o1))
        :effect (and (robot ?c1 ?o2) (not (robot ?c1 ?o1)))
)
```

Figura 10 – Acción de PDDL

Una acción en PDDL viene determinada por la declaración y su nombre [:action turn], por los parámetros de la acción [:parameters (?c1 - cell ?o1 - orientation ?o2 - orientation)], por las precondiciones necesarias que deben satisfacerse para que esta acción la tenga en cuenta el planificador [:precondition (and (robot ?c1 ?o1))] y el resultado del acción [:effect (and (robot ?c1 ?o2) (not (robot ?c1 ?o1)))]. Las variables de los parámetros, precondiciones y efectos son predicados, que vendrán en el estado inicial del problema o bien se irán creando, modificando o eliminando por las acciones. Se utiliza la palabra clave “and” como conjunción lógica de condicionales y la palabra clave “not” para eliminar predicados, sólo podrá usarse en los resultados de las acciones (bloque“:effect”).

2.3.2 Metric-FF

Metric-FF es un sistema de planificación independiente del dominio[22]. Se trata de una extensión del planificador FF combinado con ADL con variables de estado numéricas. Soporta nivel de PDDL 2.1 nivel 2. Para la búsqueda utiliza una variación de el algoritmo *hill-climbing* en el espacio de todos los estados alcanzables. La evaluación de heurísticas se realiza mediante la resolución de tareas relajadas en cada búsqueda de estado usando un algoritmo graphplan [23].

2.4 PELEA

PELEA (Planning, Execution and Learning Architecture) es una arquitectura que utiliza la planificación automática y el aprendizaje automático para resolver problemas de forma automática. Esta arquitectura está distribuida en componentes de forma que éstos pueden ser sustituidos de forma sencilla. PELEA utiliza un bucle de planificación, monitorización y ejecución de forma entrelazada [24].

Como se ha comentado en el apartado anterior, PELEA utiliza una arquitectura de componentes. Está compuesto por ocho componentes:

- Módulo de ejecución (Execution): Este módulo sirve como conector hacia el sistema que quiere ser controlado (robot, máquina, etc). Se encarga de obtener información de los sensores del sistema a controlar y de ejecutar las acciones que han sido previamente generadas por el planificador.
- Módulo de monitoreo (Monitoring): Este módulo se encarga de gestionar las comunicaciones de los diferentes nodos, así como sincronizar el proceso de planificación y ejecución. Además se encarga de monitorizar el estado del sistema.
- Módulo de soporte de decisión (Decision support): Este módulo se encarga de decidir qué predicados se deben monitorizar y cuales son sus valores válidos. Además se encarga de verificar que el estado real se corresponda con el esperado y en caso de no ser así, replanificar.
- Planificador de alto nivel (High-level replanner): Este módulo se encarga de generar el plan que solucionara el problema ya que encapsula un planificador automático independiente de dominio. Además, adapta el plan del lenguaje del planificador a XML y vice-versa.
- Planificador de bajo nivel (Low-level planner): Este módulo se encarga de traducir las acciones de alto nivel a acciones de bajo nivel.

- Traductor bajo nivel a alto nivel (LowToHigh): Este módulo se encarga de traducir el estado del problema de bajo nivel a alto nivel. También traduce los valores de los sensores en conocimiento de alto nivel.
- Aprendizaje (Learning): Este módulo se encarga de generar datos de aprendizaje a partir de los resultados de los planes ejecutados.
- Metas y generación de métricas (Goal & metric generation): Este módulo se encarga de gestionar las metas y las métricas, como utilizar subconjuntos de metas en caso de que no se puedan resolver todas o descubrir/crear nuevas metas de manera dinámica. También permite de establecer los criterios (métricas) respecto a los que se optimizan los planes.

En la Figura 11 se puede ver la arquitectura de PELEA y como se comunican sus componentes. En el diagrama se utilizan palabras como stateL, domainL, stateH, domainH… Todos los elementos acabados en "L" se refieren al elemento de bajo nivel (Low-level), por ejemplo, stateL se refiere al estado de bajo nivel y domainL se refiere al dominio de bajo nivel. Por otra parte, los elementos acabados en "H" se refieren al elemento de alto nivel (High-level), por ejemplo, stateH se refiere al estado de alto nivel y domainH se refiere al dominio de alto nivel.

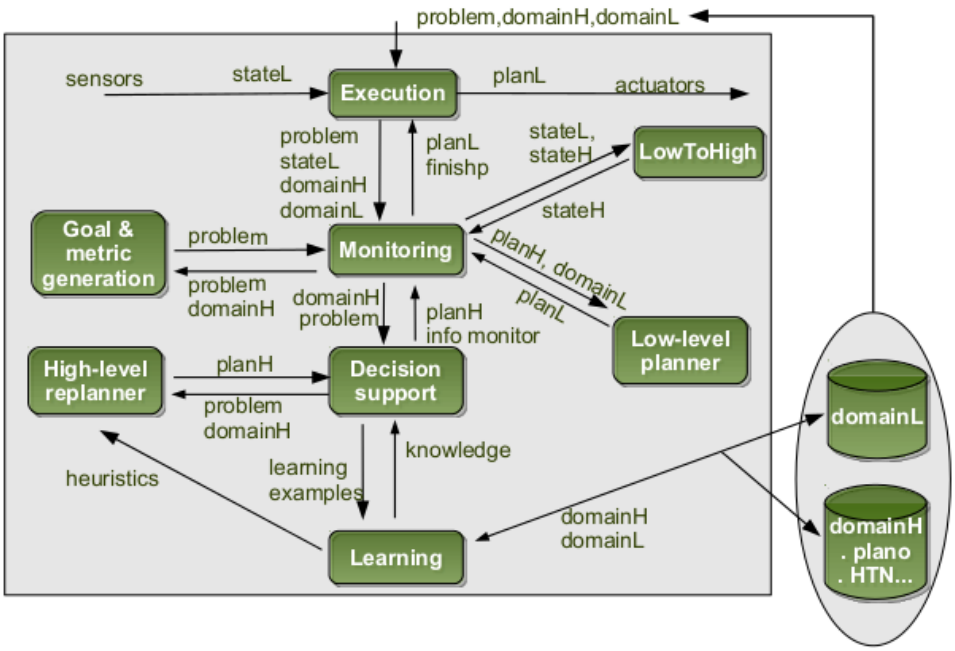


Figura 11 – Arquitectura de pelea

2.4.1 Ciclo de ejecución de PELEA

Dependiendo de las necesidades y la complejidad del problema a resolver, la arquitectura puede ser configurada con el fin de simplificar su funcionamiento. Una de estas simplificaciones corresponde con Pelea1level[25]. Pelea1level se trata de la versión básica de PELEA, consta de los nodos de Decision support, monitoring y Execution. Se ha escogido esta versión ya que no se utilizan modelos de alto y bajo nivel dentro de PELEA, ya que solo se utiliza alto nivel; tampoco se utiliza replanificación ni aprendizaje, por lo tanto, la versión básica es suficiente para planificar el problema, monitorizar el estado del mismo y al robot y enviar las acciones necesarias al controlador del robot para que las ejecute.

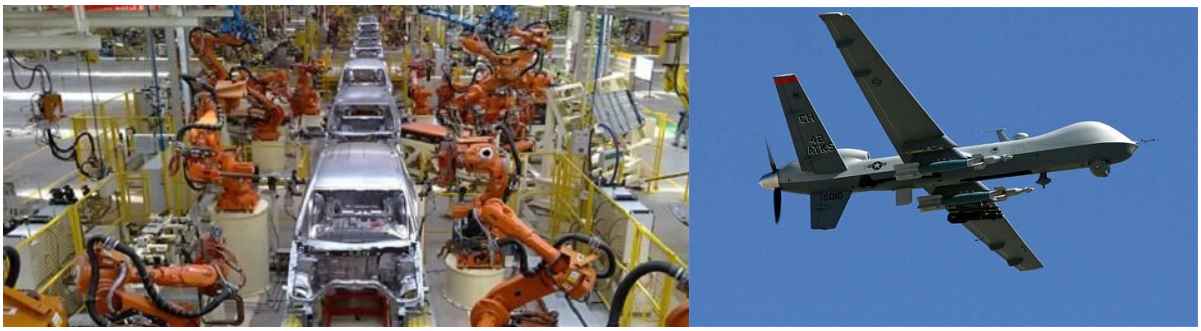
PELEA puede ser utilizado como controlador principal del sistema, pero para este trabajo se ha utilizado el controlador del robot como controlador principal, y es el que se encarga de ejecutar PELEA. Este apartado se detalla en la sección 3.3.3.

El funcionamiento de pelea1level es el siguiente, Monitoring se mantiene a la espera para registrar los otros nodos en la red y se parsean los ficheros de dominio y problemas. Tras esto, Monitoring pide a execution el estado actual del mundo y envía las metas junto con el estado a Decision Support. Decision support ejecuta un planificador para resolver el problema y una vez resuelto le envía el plan de acciones a Monitoring y despúes se envía la siguiente acción a ejecutar a Execution que se encargará de enviársela al controlador del robot. Una vez que se ha enviad la primera acción, se repite el bucle descrito en la Figura 12, que se describe a continuación. Una vez que se ha enviado la primera acción, Execution se mantiene a la espera de una contestación del sistema de control del robot; cuando la recibe, Execution envía el estado del mundo a Monitoring, éste lo compara y si el resultado es satisfactorio, se continúa con la siguiente acción que se envía a Execution, en caso contrario se vuelve a planificar mediante Decision Support. Se repite este proceso hasta que se hayan cumplido las metas.

Figura 12 – Ciclo de ejecución pelea1level

2.5 Robots

No hay una única definición de robot que satisfaga a todo el mundo, y cada uno tiene la suya. Por ejemplo Joseph Engelberger (pionero en robots industriales) una vez dijo "*I can't define a robot, but i know one when i see one*". Y cada vez es más difícil definir a un robot, ya que según avanza la tecnología, son más diferentes y diversos. El Insituto de Robótica de América (Robot Institute of America) propuso la siguiente definición en 1979: Un manipulador reprogramable, multifuncional diseñado para mover material, partes, herramientas o aparatos especializados mediante diversos movimientos programados para mejorar el rendimiento de diversas tareas. Actualmente los robots varían mucho ya que el campo ha crecido enormemente, se pueden encontrar robots fijos en cadenas de montaje (Figura 13(a)) como otros que son capaces de volar de manera autónoma (Figura 13(b)).



(a) (b)

Figura 13 – Robots industriales manipuladores en una cadena de montaje (a) MQ-9 Reaper armado en vuelo (b)

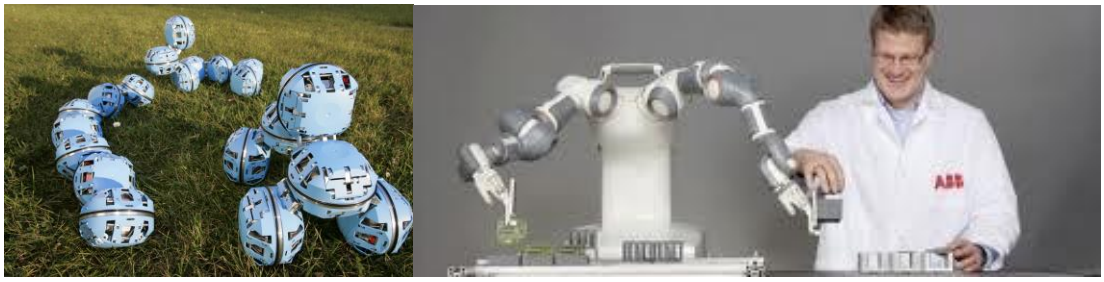
Según su función o propiedades, se pueden diferenciar seis grandes grupos:

- Robots móviles: Tienen la capacidad de moverse en su entorno y no se encuentran anclados en una localización. Por ejemplo el UAV MQ-9 (Figura 13(b)).
- Robots industriales (manipuladores): Suelen estar formados por un brazo articulado unidos a un efector final que unido a una superficie fija. Por ejemplo brazos industriales en una cadena de montaje (Figura 13(a)).
- Robots de servicio: Se trata de robots que actúan de manera semi o totalmente autónoma para realizar actividades útiles para el bienestar de los humanos, excluyendo la fabricación. Por ejemplo la aspiradora automática Roomba (Figura 14(a)).

- Robots educacionales: Permiten ayudar a aprender (generalmente a niños) o asistir a profesores en las labores de aprendizaje. Por ejemplo el robot programable para niños Play-I (Figura 14(b)).
- Robots modulares: Se tratan de robots que pueden ser unidos unos con otros para aumentar su funcionalidad, basándose en una arquitectura modular que permite acoplarlos. Por ejemplo el robot Atron (Figura 15(a)).
- Robots colaborativos: Permiten interactuar con trabajadores humanos en tareas industriales simples de manera segura y efectiva. Por ejemplo el brazo FRIDA (Figura 15(b)).

(a) (b)

Figura 14 – Robot de Servicio Roomba (a) Robot educacional Play-I (b)



(a) (b)

Figura 15 – Robot modular Atron (a) Robot colaborativo FRIDA (b)

2.5.1 Robots móviles

Se tratan de robots cuya propiedad principal es la habilidad de moverse en su entorno. Hay diversos entornos en los que son capaces de navegar como el aire (UAV), en la tierra (AGV), en el agua (AUV) e incluso en el espacio exterior (por ejemplo RRM).

El gran problema o reto de los robots móviles es lidiar con las situaciones adversas que pueden producirse cuando se mueven por entornos abiertos, este tipo de circunstancias pueden causar errores en el funcionamiento del robot y en el peor de los casos poner en riesgo la integridad del robot. Para intentar prevenir estas situaciones de riesgo o el error producido por el entorno en el que se encuentra el robot, se utilizan una gran variedad de sensores para obtener la información del entorno. Los más utilizados son:

- Odómetros: Indican la distancia recorrida por el robot.
- Giroscopios: Determina la rotación producida por el robot durante su movimiento.
- Acelerómetros: Determina la aceleración producida por el robot durante su movimiento.
- GPS (Global Position System): Determina la posición del robot en la Tierra mediante sistemas satélite.
- Sensores infrarrojos: Miden la luz infrarroja emitida por los objetos.
- Sensores ultrasónicos: Utiliza ondas de ultrasonidos para detectar obstáculos.
- Sensores de contacto: Permite saber si el sensor está tocando o siendo tocado por algo.
- Sensores de presión: Permite conocer la fuerza ejercida sobre el sensor.
- Sensores de profundidad: Permite medir a qué distancia se encuentra el robot bajo el nivel del mar.
- Cámaras: Permite capturar la imagen del entorno. Suele ser el sensor más importante.
- Laser: Permite determinar la distancia a un objeto mediante la emisión de un rayo de luz.

En este trabajo se ha usado el giroscopio, acelerómetro, sensores ultrasónicos, sensores de contacto y cámaras.

2.5.2 Robots humanoides

Un robot humanoide es un robot cuya apariencia física se asemeja a la del cuerpo humano. Existen varios motivos por los que se diseñan robots con aspecto humanoide, para el uso de herramientas humanas, para mejorar la interacción hombre-máquina, para estudiar el movimiento bípedo, para el aprendizaje…

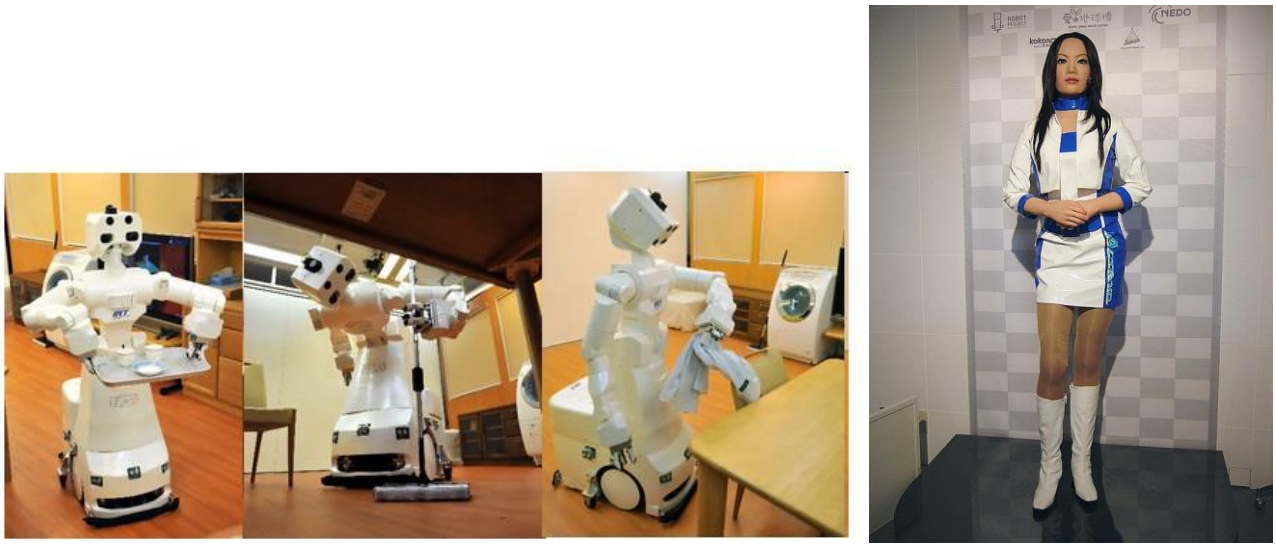
Los robots humanoides son usados como herramienta de investigación en numerosas áreas. Es necesario un conocimiento previo de la estructura y comportamiento del cuerpo humano para construir robots humanoides, aunque el intento de imitar el cuerpo humano lleva a un mejor entendimiento del mismo; por ejemplo la planificación de la posición de los pies del robot ASIMO (Figura 16(a)) para esquivar obstáculos [26]. También se utilizan en la investigación de la cognición humana, para modelar comportamientos similares a los de los humanos y entender cómo se lleva a cabo el aprendizaje; por ejemplo la investigación de la comunicación no verbal en la interacción hombre-máquina con el robot NAO[27] (Figura 16(b)). También cabe destacar la investigación en prótesis y órtesis para los seres humanos que se ha conseguido a través de la construcción de robots humanoides articulados.

(a) (b)

Figura 16 – Robot Asimo bajando escaleras (a) Robot NAO ayudando a un niño con desórdenes sociales (b)

Aparte de la investigación, también se utilizan robots humanoides para desempeñar tareas que puede realizar un humano, asistirlo y en ciertas ocasiones sustituirlo. Algunas de estas tareas son, cuidado y asistencia de personas mayores, asistente personal (Figura 16(a)), labor de recepcionista (Figura 16(b))… En teoría los robots humanoides podrían desempeñar cualquier labor que realizar un humano ya que su forma es la misma, siempre y cuando tengan el software adecuado para ello; pero desarrollar este tipo de software es realmente complejo. También se está volviendo popular

el uso de robots humanoides como herramienta de entretenimiento, por ejemplo en parques temáticos o como un juguete más.



(a) (b)

Figura 17 – AR robot realizando tareas domésticas (a) Actroid-DER 01(b)

2.5.3 Robot NAO

El robot NAO[28] desarrollado por la empresa Aldebaran Robotics, se trata de un robot humanoide bípedo con 25 grados de libertad (DoF). Los grados de libertad corresponden con las articulaciones del robot y están ubicados de la siguiente manera: cinco en cada pierna, cinco en cada brazo, uno en cada mano, uno en la pelvis y dos en la cabeza. Dispone de los siguientes dispositivos para obtener la información de su entorno: 2 dispositivos de ultrasonido, 2 cámaras, 2 bumpers, 8 sensores de fuerza, 3 botones táctiles, un giroscopio y un acelerómetro. En la ilustración 12 se puede ver la ubicación de cada uno de estos dispositivos. En la figura 18 se muestra el hardware del robot y su ubicación.

La arquitectura de software del robot se llama NaoQi y está diseñada como un sistema distribuido donde cada componente puede ser ejecutado localmente en el sistema interno del robot o ejecutado de manera distribuida entre otros sistemas mientras el Daemon NaoQi está activo en el sistema principal. La única limitación en el sistema distribuido es que el proceso principal, llamado Main Broker, tiene que estar activo en el robot Nao si se quieren usar los actuadores o sensores.

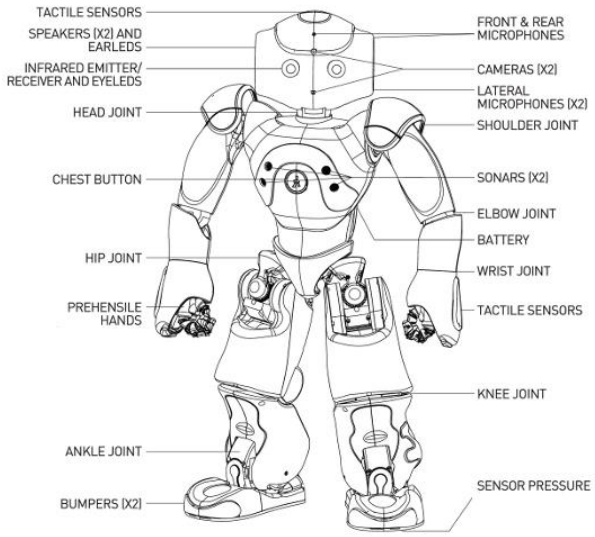


Figura 18 – Hardware del robot NAO

NaoQi está dividido en tres grandes partes NaoQi Operative System, NaoQi Library and Device Control Manager (DCM). Para comunicarse con el robot, se puede realizar mediante el DCM o con la NaoQi Library. Si se accede a través del DCM, envía órdenes directas al controlador de hardware. NaoQi Library ofrece una capa de abstracción en la cual se pueden cargar diversos módulos con funcionalidad ya desarrollada, como por ejemplo, caminar, capturar una imagen, hablar… Para el trabajo sólo se ha usado la NaoQi Library accediendo a los diversos módulos que ofrece la API, pero se ha tenido que acceder a los valores almacenados en memoria del giroscopio, acelerómetro y los sensores táctiles que son gestionados por el DCM.

Capítulo 3: Descripción del sistema

El objetivo principal de este capítulo es describir el trabajo como un sistema. En primer lugar se realiza una breve descripción de alto nivel del sistema. A continuación se presenta el análisis del sistema, en el cual se explicarán las características funcionales del sistema, el entorno operacional, la descripción de los casos de uso y los requisitos funcionales y no funcionales. Por último, se presenta el diseño detallado del sistema, para lo que se incluyen una descomposición del sistema en componentes hasta el nivel de las clases y una descripción del funcionamiento del sistema.

3.1 Introducción

Como se ha comentado anteriormente, en este trabajo consiste en el desarrollo de un sistema de control para un robot humanoide mediante la utilización de planificación automática y visión artificial, donde el robot será capaz de coger objetos y llevarlos a una posición conocida a priori. El sistema debe ser capaz de almacenar objetos mediante visión, mediante una interfaz con la que un humano sea capaz de comunicarse con el robot. Los objetos almacenados en el sistema tendrán que poderse diferenciar (al menos por el sistema) para poder ser identificados.

El sistema debe tener un planificador para poder resolver los problemas. Las acciones dadas por el planificador tendrán que ser adaptadas al robot para que pueda entenderlas. El problema debe contener toda la información del espacio por el que se puede mover el robot (espacios libres, obstáculos, objetos y lugares de almacenamiento). El robot debe ser capaz de moverse y actuar dadas las acciones del planificador, además, tendrá que ser capaz de localizar los objetos e identificarlos, recogerlos y llevarlos al destino deseado.

Para llevar la funcionalidad descrita. El sistema constará de:

- Un ordenador: encargado de ejecutar el sistema de control y PELEA.
- Un robot NAO: encargado de moverse por el entorno, proveer de sensores al sistema y capaz de coger y depositar objetos.
- Un router: encargado de comunicar el ordenador con el robot.

En la Figura 19 se presenta un esquema de los diferentes elementos hardware que serán necesarios para desplegar el sistema de control.

Figura 19 – Diagrama alto nivel del sistema

3.2 Análisis del sistema

En esta sección se presenta el análisis del sistema, comenzando por una descripción breve de las características funcionales, éstas se describirán de forma detallada en los apartados de especificación de casos de uso y en la especificación de requisitos. También se darán a conocer las restricciones del sistema (hardware y software), además de las restricciones encontradas durante la elaboración del proyecto y el entorno operacional del mismo.

3.2.1 Descripción de las características funcionales

Las características funcionales del sistema son las siguientes:

1. Poder guardar en el sistema mediante visión artificial un objeto o información necesaria para poder diferenciarlo de otros. Para que el sistema almacene esta información habrá que mostrarle el objeto al robot.
2. Reconocer objetos de pequeño tamaño usando visión artificial.
3. Recoger y depositar objetos de pequeño tamaño con ayuda de visión artificial.
4. Poder mover objetos de pequeño tamaño de un lugar a otro de manera precisa.
5. Poder obtener las acciones necesarias mediante planificación para que el robot sepa qué hacer en cada momento.
6. Modelar para un planificador, el entorno del problema, objetivos y posibles acciones que permitan resolver el problema planteado.
7. Interpretar las acciones del planificador y adaptarlas para que el robot sea capaz de realizarlas (traducción de alto nivel a bajo nivel).
8. Permitir que se almacene más de un objeto, con su respectiva recogida, desplazamiento y envío del mismo.

9. Una interfaz que permita comunicarse con el robot mediante comandos de voz y / o los botones del robot para iniciar el sistema y almacenar objetos.

3.2.2 Restricciones del sistema

En esta sección se describirán las restricciones que se han tenido en cuenta a la hora de realizar el sistema, así como las restricciones que se han encontrado durante su desarrollo y experimentación.

3.2.2.1 Restricciones de Hardware

Las restricciones de hardware del sistema corresponden a las restricciones físicas del robot NAO y de un ordenador que va a controlar y enviar las acciones al robot.

Ordenador:

- Procesador: Intel Core @500MHz o superior, o equivalente de AMD.
- Memoria RAM: 50 MB.
- Almacenamiento: 1 MB de espacio libre para los ejecutables y ficheros de configuración.
- Pantalla: resolución de pantalla de 800x600 o superior.
- Tarjeta de red, también es viable conexión Wi-Fi.

Robot NAO:

- El modelo de robot corresponde con el H25 V3.2, que posee 25 grados de libertad (DOF). Los grados de libertad corresponden con las articulaciones del robot y están ubicados de la siguiente manera: cinco en cada pierna, cinco en cada brazo, uno en cada mano, uno en la pelvis y dos en la cabeza.
- El robot dispone de dos cámaras en la cabeza con una resolución de 640x480 a 30 imágenes por segundo (fps).En la ilustración 14 se puede ver el campo de visión del robot aportada por las dos cámaras.

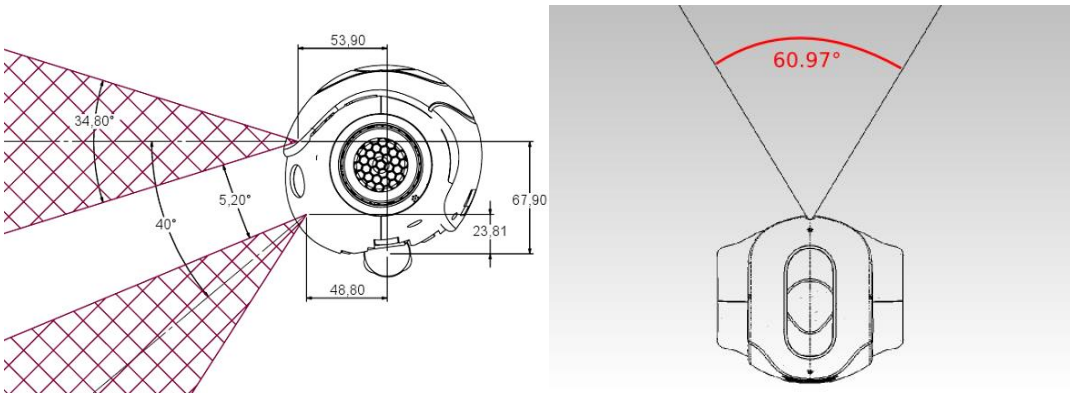


Figura 20 – Rango de las cámaras del NAO

- También dispone de un botón en el pecho y tres sensores táctiles en la cabeza.
- Se puede conectar al robot mediante WiFi o cable Ethernet.
- Finalmente, dispone de varios sensores, entre los que se encuentra un giroscopio de dos ejes (5% de precisión con una velocidad angular de ~500º/s), un acelerómetro de tres ejes (1% de precisión con una aceleración de ~2G).

Aparte del ordenador y el robot NAO, es necesario un router que provea de una red para conectar el ordenador con el robot. La conexión se podrá realizar mediante cable Ethernet o vía Wi-Fi.

3.2.2.2 Restricciones de Software

Las restricciones de software del sistema se corresponden con la versión de software de control del robot, los lenguajes de programación utilizados en el desarrollo y los sistemas operativos soportados para desplegar el sistema.

- El sistema operativo debe tener una versión del planificador Metric-FF, que se pueda ejecutar. Para compilar Metric-FF, requiere de gcc, bison y lex.
- El sistema se podrá ejecutar en cualquier distribución de Linux y MacOS compatible con las restricciones de software que se detallarán a continuación.
- La versión de software del robot corresponde con NAOqi 1.14 o superior.
- Para el desarrollo del sistema se ha utilizado Python 2.7. Junto con las librerías de OpenCV, Numpy 1.8.1(Dependencia de OpenCV), el framework PyQt4 y las librerías de NAOqi 1.14 para Python acordes al sistema operativo
- El sistema operativo deberá tener Java 1.6 o superior.

- Para el desarrollo de visión artificial se han usado las librerías de OpenCV, ya que ofrece versiones estables, funcionales y es la más usada en este tipo de proyectos. Para la visualización de la cámara del robot se ha utilizado el framework PyQt4, ya que es la recomendada en la documentación para la API del robot NAOqi 1.14.
- PELEA está desarrollado en Java, y las modificaciones necesarias para integrar el Control con PELEA se han realizado en Java y Python.
- Para poder ejecutar el sistema, el sistema operativo tendrá que tener instalado Python 2.7, las librerías de Python de OpenCV, Numpy (Dependencia de OpenCV), el framework PyQt4 y las librerías de NAOqi 1.14 para Python acordes al sistema operativo. También requiere tener instalado Java 1.6 o superior.

3.2.2.3 Restricciones adicionales encontradas

- Lenguaje para desarrollar el sistema de control: Java se descartó ya que la API del robot en este lenguaje estaba incompleta, ya que faltaban algunas funciones necesarias para el sistema. C++ se descartó ya que el desarrollo en este lenguaje resultaría más lento que en Python. Se decidió utilizar Python por ser un lenguaje de programación que hacía más ágil el desarrollo del sistema, principalmente al interactuar con la API del robot (NAOqi 1.14) y en el desarrollo de visión artificial.
- Iluminación: La iluminación debe ser constante durante toda la ejecución del sistema, ya que en caso contrario, el módulo de visión artificial puede dar resultados erróneos debido a la variación de luz sobre el entorno.

3.2.3 Entorno operacional

En este apartado se realiza una descripción del entorno operacional del sistema, que incluye: software, hardware y del problema.

3.2.3.1 Entorno operacional software

El entorno operacional de software viene dado por el sistema operativo a utilizar y las librerías necesarias para ejecutar el sistema:

- Sistemas operativos en los que se ha probado el sistema: Ubuntu 14.04 LTS y OS X Mavericks 10.9.4.
- El planificador automático ha sido utilizado por PELEA es Metric-FF.
- Se ha utilizado la máquina virtual de Java 1.7.

- Las versiones de software que se han usado son las siguientes: Python 2.7, OpenCV 2.4.9, Numpy 1.8.1, PyQt4 4.11.1, NAOqi 1.14.

Adicionalmente, durante el desarrollo se han utilizado los siguientes programas para realizar pruebas sin el robot:

- Webots for NAO 7.4.3: Permite simular al robot y utilizarlo en un entorno controlado.
- Choregraphe: Permite grabar y ejecutar comportamientos para el robot y ver su estado.

3.2.3.2 Entorno operacional hardware

El entorno operacional hardware viene dado por las piezas que son necesarias para hacer funcionar el sistema:

- Se ha utilizado un robot NAO H25 V3.2 de la empresa Aldebaran Robotics ©.
- Se ha utilizado un portátil con las características de descritas en 3.2.2.1 y con el entorno descrito en 3.2.3.1, encargado de ejecutar PELEA y Control.
- Un router que sea capaz de establecer una conexión entre el robot y el ordenador portátil mediante una red local.

3.2.3.3 Entorno operacional del problema

El entorno operacional del problema corresponde con la ubicación, luminosidad y otros agentes externos:

- El suelo debe permitir caminar adecuadamente al robot, sin que éste pueda tropezarse por desniveles.
- El entorno del problema debe tener una luminosidad adecuada para la cámara del robot, ya que si es un entorno oscuro, muy claro o el objeto se encuentra a contra-luz el robot no será capaz de almacenar o reconocer adecuadamente el objeto.
- El objeto a recoger debe tener un tamaño pequeño, pesar poco y que el robot sea capaz de recogerlo, para el proyecto se han utilizado calcetines.

3.2.4 Especificación de casos de uso

En este apartado se va a describir de manera detallada la especificación de casos de uso del sistema, comenzando con una descripción de los actores y sus interacciones, mostrándose gráficamente con un diagrama. Después se verá uno a uno cada caso de uso del sistema para poder establecer los requisitos en base a los casos de uso.

3.2.4.1 Descripción de los actores

En esta sección se presenta el diagrama de casos de uso del sistema. En el diagrama presentado en la Ilustración 15, se pueden observar tres actores: el usuario, el robot NAO y la arquitectura PELEA.

- Usuario: Se corresponde con el usuario que indicará al robot la tarea a realizar, así como los diferentes objetos que tendrá que recoger.
- Control (actor virtual): Se corresponde con el sistema a desarrollar en este trabajo. Se encargará de coordinar la interacción entre los otros actores, del procesamiento parcial de la información obtenida a través de los diferentes sensores del robot y de la definición de las acciones de bajo nivel que deberá ejecutar el robot.
- Robot: Se corresponde con el dispositivo que será controlado por el sistema para la realización de las acciones.
- PELEA: Se corresponde con el sistema de planificación automática que se encargará de generar una solución al problema y controlar el flujo de ejecución de las acciones de alto nivel que ejecutará el robot.

En el diagrama de casos de uso, presentado en la Figura 15, se puede observar que todos los actores externos se comunican entre sí a través del actor virtual que representa el sistema.

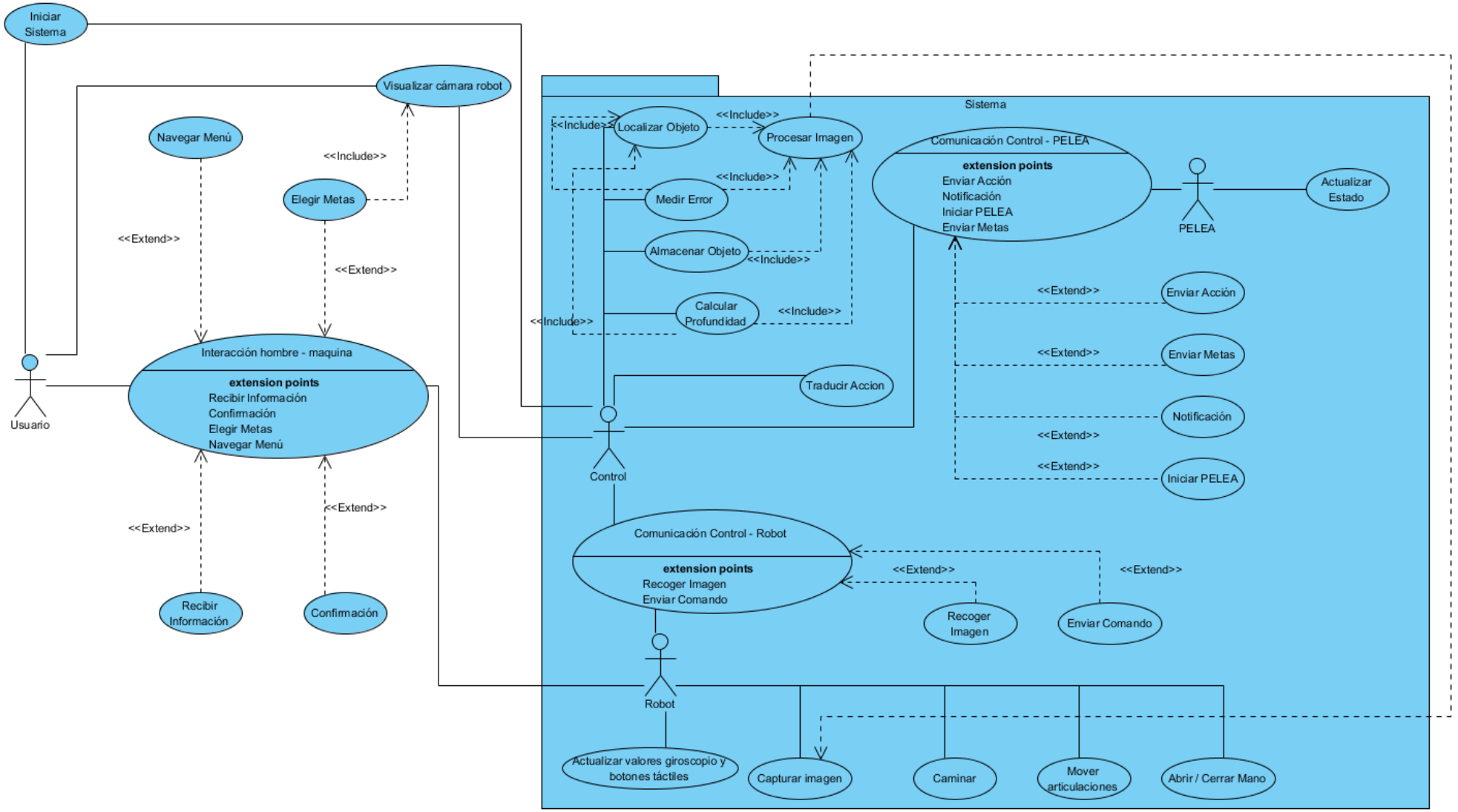


Figura 21 – Diagrama de casos de uso del sistema

3.2.4.2 Descripción de los atributos de los casos de uso

Para la realización de la descripción textual de los distintos casos de uso, se han seleccionado una serie de atributos que describen cada uno de los casos de uso. A continuación se realiza una descripción del significado de cada uno de los atributos utilizados para la descripción de los casos de uso.

- Código: Identificación unívoca abreviada del caso de uso, se construye mediante CU seguido de un - y de tres dígitos. Por ejemplo CU-001.
- Nombre: Identificación extendida del caso de uso.
- Actores: Conjunto de entidades que interactúan con el caso de uso. El caso de uso representa una funcionalidad demandada por un actor.
- Descripción: Se realiza una descripción básica de la funcionalidad o funcionalidades del caso de uso.
- Precondiciones y poscondiciones: Se realiza una descripción de las condiciones que deben cumplirse para poder realizar una operación, y el estado en el que queda el sistema tras realizar una operación.
- Escenario: Se realiza una descripción básica de las acciones que se ejecutaran paso a paso en el caso de uso.

3.2.4.3 Descripción textual de los casos de uso

En el siguiente apartado se va informar de cada caso de uso de manera detallada, sin incluir los casos de uso "Interacción hombre – máquina", "Comunicación Control – PELEA" y "Comunicación Control – Robot" ya que se trata de la unión de los casos de uso que extienden su funcionalidad.

Código	**CU-01**
Nombre	Iniciar Sistema
Actores	Usuario, Control
Descripción	Permite al usuario iniciar el sistema
Precondiciones	• El robot debe estar encendido y conectado a la red. • El usuario debe estar conectado a la misma red que el robot. • El fichero de configuración deberá tener los parámetros adecuados.
Poscondiciones	El sistema se iniciará.
Escenario	El usuario inicia el sistema, Control se iniciará y se comunicará con el robot.

Código	**CU-02**
Nombre	Navegar Menú
Actores	Usuario, Robot
Descripción	Permite al usuario seleccionar qué es lo que quiere hacer con el robot.
Precondiciones	• El robot debe estar encendido y conectado a la red. • El usuario debe estar conectado a la misma red que el robot. • Control debe estar funcionando. • El robot ha informado al usuario en qué menú está y qué opciones tiene.
Poscondiciones	• El usuario ha elegido una opción de las que el robot le ha ofrecido y el sistema transitará a la opción seleccionada.
Escenario	El robot mediante audio comunica al usuario el menú en el que se encuentra y qué posibilidades tiene, el usuario presiona uno de los botones táctiles situado en la cabeza del robot, el robot recibe la señal y el sistema transitará a la opción seleccionada.

Código	**CU-03**
Nombre	Elegir Metas
Actores	Usuario, Robot
Descripción	Permite al usuario almacenar una meta en el sistema.
Precondiciones	• El robot debe estar encendido y conectado a la red. • El usuario debe estar conectado a la misma red que el robot. • Control debe estar funcionando. • El robot ha informado al usuario en qué menú está y qué opciones tiene. • Se muestra la cámara del robot en la pantalla del ordenador.
Poscondiciones	• Una nueva meta (objeto) se ha añadido al problema.
Escenario	El robot mediante audio comunica al usuario el menú en el que se encuentra y qué posibilidades tiene, el usuario presiona uno de los botones táctiles situado en la cabeza del robot para almacenar el objeto en el sistema, el robot recibe la señal y el sistema almacenará el objeto para el problema.

Código	**CU-04**
Nombre	Recibir Información
Actores	Usuario, Robot
Descripción	Permite al usuario conocer información del sistema mediante los altavoces del robot.
Precondiciones	• El robot debe estar encendido y conectado a la red. • El usuario debe estar conectado a la misma red que el robot. • Control debe estar funcionando.
Poscondiciones	• El robot ha informado al usuario mediante audio del estado del sistema.
Escenario	El robot mediante audio comunica al usuario el menú en el que se encuentra y qué posibilidades tiene.

Código	**CU-05**
Nombre	Confirmación
Actores	Usuario, Robot
Descripción	Da la posibilidad al usuario de confirmar o denegar la acción que el robot le indica.
Precondiciones	• El robot debe estar encendido y conectado a la red. • El usuario debe estar conectado a la misma red que el robot. • Control debe estar funcionando. • El robot ha informado al usuario qué acción va a realizar y le pide su confirmación.
Poscondiciones	• El robot recibe la señal de confirmación o denegación de la acción anunciada previamente, y se lo comunica al sistema.
Escenario	El robot informa al usuario qué acción se va a realizar y espera confirmación, el usuario presiona uno de los botones táctiles situados en la cabeza del robot para confirmar, denegar o repetir la información. Una vez confirmado o denegado el robot recibe la señal y se lo comunica al sistema.

Código	**CU-06**
Nombre	Visualizar cámara robot
Actores	Usuario, Control
Descripción	Permite al usuario conocer la imagen que recibe el robot.
Precondiciones	• El robot debe estar encendido y conectado a la red. • El usuario debe estar conectado a la misma red que el robot. • Control debe estar funcionando.
Poscondiciones	• El usuario puede ver la imagen que recibe el robot.
Escenario	Se muestra un pequeño widget en la pantalla con la imagen que recibe el robot en modo de vídeo.

Código	**CU-07**
Nombre	Obtener valores sensores giroscopio y botones táctiles
Actores	Robot
Descripción	Permite conocer la información de los sensores del robot (giroscopio y botones táctiles) al sistema.
Precondiciones	• El robot debe estar encendido y conectado a la red. • El usuario debe estar conectado a la misma red que el robot. • Control debe estar funcionando. • Recibir una petición de Control.
Poscondiciones	• El sistema dispone los valores de los sensores del robot.
Escenario	El sistema requiere el valor de uno o varios sensores del robot, el robot recibe la petición y el robot facilita la información de sus sensores al sistema.

Código	**CU-08**
Nombre	Capturar Imagen
Actores	Robot
Descripción	Permite al sistema capturar la imagen de una de las cámaras del robot.
Precondiciones	• El robot debe estar encendido y conectado a la red. • El usuario debe estar conectado a la misma red que el robot. • Control debe estar funcionando. • Recibir una petición de Control.
Poscondiciones	• El sistema dispone de la imagen de una de las cámaras del robot.
Escenario	El sistema requiere una imagen de una de las cámaras del robot, el robot recibe la petición y el robot facilita la imagen al sistema.

Código	**CU-09**
Nombre	Caminar
Actores	Robot
Descripción	Permite al sistema hacer caminar al robot en una dirección (x, y) durante un período de tiempo.
Precondiciones	• El robot debe estar encendido y conectado a la red. • El usuario debe estar conectado a la misma red que el robot. • Control debe estar funcionando. • Recibir una petición de Control. • El robot tiene que estar de pie. • El robot tiene que tener rigidez suficiente en las piernas.
Poscondiciones	• El robot ha caminado en la dirección (x, y) y período de tiempo indicados.
Escenario	El sistema requiere mover al robot, el robot recibe la petición y el robot se mueve con los datos recibidos.

Código	**CU-10**
Nombre	Mover articulaciones
Actores	Robot
Descripción	Permite al sistema mover las articulaciones indicadas del robot (x) grados durante un período de tiempo.
Precondiciones	• El robot debe estar encendido y conectado a la red. • El usuario debe estar conectado a la misma red que el robot. • Control debe estar funcionando. • Recibir una petición de Control. • El robot tiene que tener rigidez suficiente en las articulaciones a mover.
Poscondiciones	• El robot ha movido las articulaciones en los grados y período de tiempo indicados.
Escenario	El sistema requiere mover una o varias articulaciones del robot, el robot recibe la petición y el robot mueve las articulaciones del robot con los datos recibidos.

Código	**CU-11**
Nombre	Abrir / Cerrar Mano
Actores	Robot
Descripción	Permite al sistema abrir y cerrar las manos del robot.
Precondiciones	• El robot debe estar encendido y conectado a la red. • El usuario debe estar conectado a la misma red que el robot. • Control debe estar funcionando. • Recibir una petición de Control.
Poscondiciones	• El robot ha abierto o cerrado sus mano.
Escenario	El sistema requiere abrir o cerrar las manos del robot, el robot recibe la petición y el robot realiza la acción recibida.

Código	**CU-12**
Nombre	Recoger Imagen
Actores	Control, Robot
Descripción	Permite facilitar a Control la imagen capturada por el robot.
Precondiciones	• El robot debe estar encendido y conectado a la red. • El usuario debe estar conectado a la misma red que el robot. • El Control debe estar funcionando.
Poscondiciones	• Control dispone de la imagen capturada por el robot.
Escenario	Control requiere la imagen del robot para procesarla, el robot recibe la petición y el robot le envía la imagen.

Código	**CU-13**
Nombre	Enviar comando
Actores	Control, Robot
Descripción	Permite a Control obtener información del robot, hacer que éste camine o mueva articulaciones.
Precondiciones	• El robot debe estar encendido y conectado a la red. • El usuario debe estar conectado a la misma red que el robot. • El Control debe estar funcionando.
Poscondiciones	• El robot ha recibido la petición y la realiza.
Escenario	Control necesita que el robot realice una acción, el robot la recibe y éste la realiza.

Código	**CU-14**
Nombre	Traducir acción
Actores	Control
Descripción	Permite a Control adaptar la acción del planificador (alto nivel) y descomponerla en comandos que el robot pueda entender (bajo nivel).
Precondiciones	• Control debe estar funcionando. • PELEA debe estar funcionando. • Haber recibido una acción de PELEA.
Poscondiciones	• Acción descompuesta en comandos para que el robot pueda realizar la acción del planificador.
Escenario	PELEA envía una acción a Control, Control la recibe e identifica, en función de la acción recibida, la descompone en comandos y los adapta en función de la situación actual del robot.

Código	**CU-15**
Nombre	Procesar Imagen
Actores	Control
Descripción	Permite a Control tratar una imágen, haciendo transformaciones de color, filtros, detección de formas y colores.
Precondiciones	• Control debe estar funcionando. • Una imagen.
Poscondiciones	• Imagen procesada.
Escenario	Control recibe una imagen para procesar, se aplican transformaciones de color de BGR a RGB y luego a HSV; después se difumina la imagen, se aplican filtros de color y de formas.

Código	**CU-16**
Nombre	Localizar objeto
Actores	Control
Descripción	Permite a Control localizar un objeto dado mediante su color y su área a partir de una imagen.
Precondiciones	• Control debe estar funcionando. • Una imagen.
Poscondiciones	• Objeto localizado con coordenadas en caso de que haya sido identificado.
Escenario	Control recibe una imagen y tiene que identificar si la imagen tiene el objeto, para ello, procesará la imagen (CU-15) y en caso de que identifique el objeto tiene que localizar en qué posición de la imagen se encuentra.

Código	**CU-17**
Nombre	Medir Error
Actores	Control
Descripción	Permite a Control estimar la diferencia de posición entre un objeto localizado y el centro de la imagen dado un margen de confianza, para corregir la trayectoria del robot.
Precondiciones	• Control debe estar funcionando. • Una imagen.
Poscondiciones	• Código de error.
Escenario	Control recibe una imagen, obtiene la posición del objeto en la imagen (CU-16), calcula la diferencia absoluta respecto al centro de la imagen, y si es superior al límite de confianza, devuelve el tipo de error para aplicar las correcciones necesarias.

Código	**CU-18**
Nombre	Calcular Profundidad
Actores	Control
Descripción	Permite a Control estimar la distancia entre el robot y el objeto a partir de una imagen, para decidir qué acción de aproximación realizar.
Precondiciones	• Control debe estar funcionando. • Una imagen.
Poscondiciones	• Código de proximidad al objeto.
Escenario	Control recibe una imagen, obtiene la posición del objeto en la imagen (CU-16), y se calcula la distancia al objeto mediante la posición del cuello del robot y qué tipo de cámara se está usando en el robot, finalmente se devuelve el código de proximidad.

Código	**CU-19**
Nombre	Almacenar objeto
Actores	Control
Descripción	Permite a Control almacenar un objeto en el sistema, mediante su detección de color.
Precondiciones	• Control debe estar funcionando. • Una imagen.
Poscondiciones	• Objeto almacenado en el sistema.
Escenario	Control recibe una imagen, y la procesa (CU-15), después, obtiene el color predominante de la imagen dentro de una circunferencia con centro en el centro de la imagen, obtiene el rango de color predominante y aplica un margen de error. El color se guarda utilizando como información el rango de color predominante.

Código	**CU-20**
Nombre	Iniciar PELEA
Actores	Control, PELEA
Descripción	Permite a Control iniciar PELEA con la configuración adecuada.
Precondiciones	• Control debe estar funcionando. • PELEA no debe estar funcionando. • Ficheros de configuración de PELEA.
Poscondiciones	• PELEA se ha iniciado y lo notifica a Control.
Escenario	Control abre un socket para comunicarse con PELEA y ejecuta un script con la configuración necesaria para ejecutar PELEA con los módulos necesarios, una vez que PELEA se ha inicializado, le notifica al subsistema de comunicación a través del socket.

Código	**CU-21**
Nombre	Enviar Acción
Actores	Control, PELEA
Descripción	Permite que PELEA envíe la acción a realizar obtenida del planificador para procesarla en Control.
Precondiciones	• Control debe estar funcionando. • PELEA debe estar funcionando. • Comunicación abierta entre Control y PELEA. • Una acción del planificador.
Poscondiciones	• Control ha recibido la próxima acción a realizar.
Escenario	PELEA obtiene la próxima acción que tendrá que realizar el robot, la envía a través de un socket a Control y éste la recibe, PELEA se mantiene esperando hasta que la acción se ha realizado (con o sin éxito).

Código	**CU-22**
Nombre	Enviar Metas
Actores	Control, PELEA
Descripción	Permite a Control enviar las metas a PELEA para que conozca cuales son los objetivos a resolver y se incluya en el problema a resolver.
Precondiciones	• Control debe estar funcionando. • PELEA debe estar funcionando. • Comunicación abierta entre Control y PELEA. • Metas a añadir.
Poscondiciones	• PELEA ha recibido las metas y las incluye en el estado del problema a resolver.
Escenario	Control ya conoce los objetos a recoger, los adapta para el lenguaje del planificador y los envía en forma de metas a PELEA. PELEA los incluye en el estado del problema a resolver.

Código	**CU-23**
Nombre	Notificación
Actores	Control, PELEA
Descripción	Permite a Control y a PELEA notificar uno al otro que se ha recibido correctamente el mensaje, se ha procesado y se puede enviar otro.
Precondiciones	• Control debe estar funcionando. • PELEA debe estar funcionando. • Comunicación abierta entre Control y PELEA. • Mensaje enviado al receptor.
Poscondiciones	• El emisor conoce que su mensaje se ha procesado y ya puede enviar el próximo.
Escenario	El emisor (PELEA o Control) envía un mensaje, se mantiene a la espera hasta que el receptor envía un mensaje de notificación, indicando si se ha realizado con éxito o no la operación del emisor. El emisor conoce el resultado de su mensaje enviado y que el receptor está libre de nuevo y puede recibir un nuevo mensaje.

Código	**CU-24**
Nombre	Actualizar estado
Actores	PELEA
Descripción	Permite a PELEA establecer el nuevo estado del problema.
Precondiciones	• PELEA debe estar funcionando. • Estado del problema.
Poscondiciones	• Nuevo estado del problema.
Escenario	El estado actual del problema ha cambiado y hay que actualizarlo dado un estado del problema. Una vez generado el nuevo estado, se establece como estado actual.

3.2.5 Especificación de requisitos

En este apartado se va a describir de manera detallada la especificación de requisitos del sistema. Para la tarea elaboración de la especificación de requisitos se mostrará en detalle cada requisito y al final de este apartado se podrá ver una matriz de trazabilidad entre los casos de uso y requisitos. Para definir los atributos de la especificación de requisitos, se ha utilizado la descrita por el IEEE[29].Acorde a esta definición, toda especificación de requisitos deberá ser:

- Correcta: Cada requisito de software reflejará una necesidad real del sistema.
- No ambigua: Cada requisito sólo podrá tener una interpretación posible.
- Completa: El documento reflejará todos los requisitos de software significativos.
- Consistente: Los requisitos no generarán conflictos entre ellos.
- Ordenada en cuanto importancia y/o estabilidad: Cada requisito indicará su importancia y/o su estabilidad.
- Verificable: para cada requisito existirá un proceso por el cual se pueda comprobar que el software satisface dicho requisito.
- Modificable: La estructura de la especificación permitirá realizar cambios a los requisitos de una manera simple, completa y consistente.
- Trazabilidad: El origen de cada requisito será claro y podrá ser reflejado con facilidad.

3.2.5.1 Descripción de los atributos de los requisitos

Para la realización de la descripción textual de los distintos requisitos que han sido identificados, se han seleccionado una serie de atributos que describen cada uno de los requisitos. A continuación se realiza una descripción del significado de cada uno de los atributos utilizados para su descripción:

- Código: Identificación unívoca abreviada del requisito, se construye mediante el código del requisito seguido de un - y de tres dígitos. Los requisitos serán divididos en funcionales y no funcionales y sus códigos son RF para los requisitos funcionales y RNF para los requisitos no funcionales. Por ejemplo RF-01.
- Nombre: Identificación extendida del requisito.
- Descripción: Se realiza una descripción básica del requisito que ha sido identificado.
- Fuente: Indica a través de que fuente ha sido identificado el requisitos. Normalmente este valor se corresponderá con uno o varios códigos de los casos de uso.

- Necesidad: Determina el grado de implementación del requisito. Los valores que puede tomar este atributo son los siguientes:
    - Esencial: El requisito tiene que ser implementado.
    - Deseable: Es preferible implementar el requisito, pero no es obligatorio.
    - Opcional: El requisito se podrá implementar, pero no es importante ni obligatorio.
- Prioridad: Define la importancia del requisito, de forma que permita definir el orden en el cual serán incluido en el proceso de diseño y el orden de implementación. Los valores que puede tomar este atributo son los siguientes:
    - Alta: El requisito debe ser implementado en las fases iniciales del desarrollo.
    - Media: El requisito debe ser implementado una vez que hayan sido implementados los requisitos de prioridad alta.
    - Baja: El requisito debe ser implementados en las fases finales del desarrollo. Estos requisitos no influirán en el correcto funcionamiento del sistema.
- Estabilidad: Define la estabilidad del requisitos durante la vida útil del software. Esto implica si el requisito podrá ser o no modificado durante el ciclo del vida. Los valores que puede tomar este atributo son los siguientes:
    - Estable: El requisito no puede variar durante el ciclo de vida del sistema.
    - Inestable: El requisito puede variar a lo largo de la ciclo de vida del sistema.
- Verificabilidad: Define el grado de verificabilidad de un requisito, es decir indica en qué grado es posible comprobar que el requisito se ha incorporado en el sistema desarrollado. Los valores que puede tomar este atributo son los siguientes:
    - Alta: Se puede verificar que el requisito ha sido implementado en el sistema. Este tipo de requisitos se corresponden con las funcionalidades básicas del sistema.
    - Media: Se puede verificar que el requisito ha sido implementado en el sistema. Pero requiere de una comprobación compleja o del código fuente del sistema.
    - Baja: Es difícil verificar si el requisito ha sido implementado en el sistema o en algunos casos no es posible.

3.2.5.2 Requisitos funcionales

En esta sección se describen los requisitos relacionados con la funcionalidad del sistema.

Código	RF-01	Fuente	**CU-[15-18]**
Nombre	Reconocer objetos		
Descripción	El sistema deberá ser capaz de reconocer objetos		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-02	Fuente	**CU-10,CU-11,CU-16**
Nombre	Recoger objetos		
Descripción	El sistema deberá ser capaz de recoger objetos		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-03	Fuente	**CU-09,CU-11**
Nombre	Transportar objetos		
Descripción	El sistema deberá ser capaz de transportar objetos		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-04	Fuente	**CU-10,CU-11,CU-16**
Nombre	El sistema deberá ser capaz de depositar objetos		
Descripción	El sistema deberá ser capaz de transportar objetos		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-05	Fuente	**CU-09,CU-10,CU-11, CU-[14-21],CU-24**
Nombre	Recolección autónoma		
Descripción	El sistema deberá ser capaz de recolectar objetos de manera autónoma		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

3.2.5.2.1 Visión

Esta subsección describe en detalle los requisitos para la funcionalidad de visión.

Código	RF-06	Fuente	**CU-[16-19]**
Nombre	Identificar objetos		
Descripción	El sistema deberá ser capaz de identificar objetos mediante visión artificial		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-07	Fuente	**CU-16, CU-17,CU-18**
Nombre	Localizar objetos		
Descripción	El sistema deberá ser capaz de localizar objetos mediante visión artificial		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-08	Fuente	**CU-[15-19]**
Nombre	Procesar imagen		
Descripción	El sistema deberá ser capaz de procesar imágenes, pudiendo realizar modificaciones pixel a pixel y cambiar el modelo de color de la imagen.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-09	Fuente	**CU-12,CU-13, CU-[15-19]**
Nombre	Capturar imagen		
Descripción	El sistema deberá ser capaz de capturar imágenes de las cámaras del robot		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-10	Fuente	**CU-19**
Nombre	Almacenar objeto		
Descripción	El sistema deberá ser capaz de almacenar un objeto mediante visión artificial		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

3.2.5.2.2 Movimiento

Esta subsección describe en detalle los requisitos para la funcionalidad de movimiento.

Código	RF-11	Fuente	**CU-09,CU-13**
Nombre	Desplazamiento		
Descripción	El sistema deberá ser capaz de mover al robot en los ejes (x,y)		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-12	Fuente	**CU-10,CU-13**
Nombre	Movimiento articulaciones		
Descripción	El sistema deberá ser capaz de mover cualquiera de las articulaciones del robot		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-13	Fuente	**CU-09,CU-11**
Nombre	Carga		
Descripción	El sistema deberá ser capaz de hacer que el robot transporte objetos		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-14	Fuente	**CU-09**
Nombre	Corrección error de movimiento		
Descripción	El sistema deberá ser capaz de corregir el error de los movimientos del robot		
Necesidad	Deseable	**Prioridad**	Media
Estabilidad	Inmutable	**Verificabilidad**	Media

3.2.5.2.3 Sensado

Esta subsección describe en detalle los requisitos para la funcionalidad de sensado.

Código	RF-15	Fuente	**CU-07, CU-13, CU-18**
Nombre	Posición articulaciones		
Descripción	El sistema deberá ser capaz de conocer la posición de todas las articulaciones del robot		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-16	Fuente	**CU-07, CU-13**
Nombre	Valores sensores		
Descripción	El sistema deberá ser capaz de conocer os valores de los sensores del robot		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

3.2.5.2.4 Planificación

Esta subsección describe en detalle los requisitos para la funcionalidad de planificación.

Código	RF-17	Fuente	**CU-20,CU-22**
Nombre	Interpretar problema		
Descripción	El sistema deberá ser capaz de interpretar el problema propuesto por el usuario		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-18	Fuente	**CU-20,CU-24**
Nombre	Resolver problema		
Descripción	El sistema deberá ser capaz de resolver el problema propuesto mediante planificación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-19	Fuente	**CU-20,CU-21**
Nombre	Acciones solución		
Descripción	El sistema deberá ser capaz de descomponer la resolución del problema en acciones		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-20	Fuente	**CU-14,CU-21**
Nombre	Interpretar acciones		
Descripción	El sistema deberá ser capaz de interpretar las acciones del planificador y adaptarlas para el robot		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-21	Fuente	**CU-20,CU-22**
Nombre	Modificar problema		
Descripción	El sistema deberá ser capaz de permitir al usuario modificar con facilidad el problema a resolver		
Necesidad	Deseable	**Prioridad**	Media
Estabilidad	Inmutable	**Verificabilidad**	Media

3.2.5.2.5 Comunicación

Esta subsección describe en detalle los requisitos para la funcionalidad de comunicación.

Código	RF-22	Fuente	**CU-[01-06]**
Nombre	Interfaz hombre-maquina		
Descripción	El sistema deberá tener una interfaz de comunicación entre el robot y el usuario		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-23	Fuente	**CU-[02-05]**
Nombre	Introducir metas		
Descripción	El sistema deberá facilitar al usuario una manera de introducir los objetivos a resolver del problema		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-24	Fuente	**CU-06, CU-[09-14]**
Nombre	Comunicación robot - control		
Descripción	El sistema deberá ser capaz de controlar y comunicarse con el robot		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-25	Fuente	**CU-[20-23]**
Nombre	Comunicación control - planificación		
Descripción	El sistema deberá ser capaz de comunicarse e interpretar los resultados del planificador		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RF-26	Fuente	**CU-[01-05]**
Nombre	Comunicación lenguaje natural		
Descripción	El sistema deberá ser capaz de ofrecer al usuario la posibilidad de interactuar mediante lenguaje natural		
Necesidad	Opcional	**Prioridad**	Baja
Estabilidad	Mutable	**Verificabilidad**	Alta

3.2.5.3 Requisitos no-funcionales

En esta sección se describen los requisitos que no están relacionados con la funcionalidad del sistema.

Código	RNF-01	Fuente	**CU-[01-19]**
Nombre	Seguridad		
Descripción	El sistema no pondrá en riesgo la integridad del robot ni del usuario durante la resolución del problema		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

Código	RNF-02	Fuente	**CU-[01-19]**
Nombre	Autonomía		
Descripción	El robot deberá permanecer encendido durante todo el tiempo de ejecución del problema		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inmutable	**Verificabilidad**	Alta

La tabla 1 muestra la matriz de trazabilidad, que gráficamente muestra cómo los casos de uso son cubiertos por los requisitos.

CU / RQ	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
RF-01															X	X	X	X						
RF-02										X	X					X								
RF-03									X		X													
RF-04										X	X					X								
RF-05									X	X	X			X	X	X	X	X	X	X	X			
RF-06			X													X	X	X	X					
RF-07																X	X	X						
RF-08															X	X	X	X	X					
RF-09								X				X	X		X	X	X	X	X					
RF-10																			X					
RF-11									X				X											
RF-12										X			X											
RF-13									X		X													
RF-14									X															
RF-15							X						X					X						
RF-16							X						X											
RF-17																				X				
RF-18																				X				X
RF-19																				X	X	X		
RF-20														X							X			
CU / RQ	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**

CU / RQ	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
RF-21																				X		X		
RF-22	X	X	X	X	X	X																		
RF-23		X	X	X	X																			
RF-24						X			X	X	X	X	X											
RF-25																				X	X	X	X	
RF-26	X	X	X	X	X																			
RNF-01	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					
RNF-02	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					

Tabla 1 – Matriz de trazabilidad Casos de uso - Requisitos

3.3 Diseño del sistema

En este apartado se presenta el diseño detallado del sistema. Para ello se presentará una visión general de los subsistemas que componen el sistema y posteriormente se realizará una descomposición en componentes de los diferentes subsistemas que componen el sistema.

3.3.1 Arquitectura del sistema

El sistema está compuesto por 4 subsistemas: Control, Robot, PELEA y OpenCV. En la Figura número 22 se pueden ver los subsistemas que componen al sistema y como se comunican entre ellos.

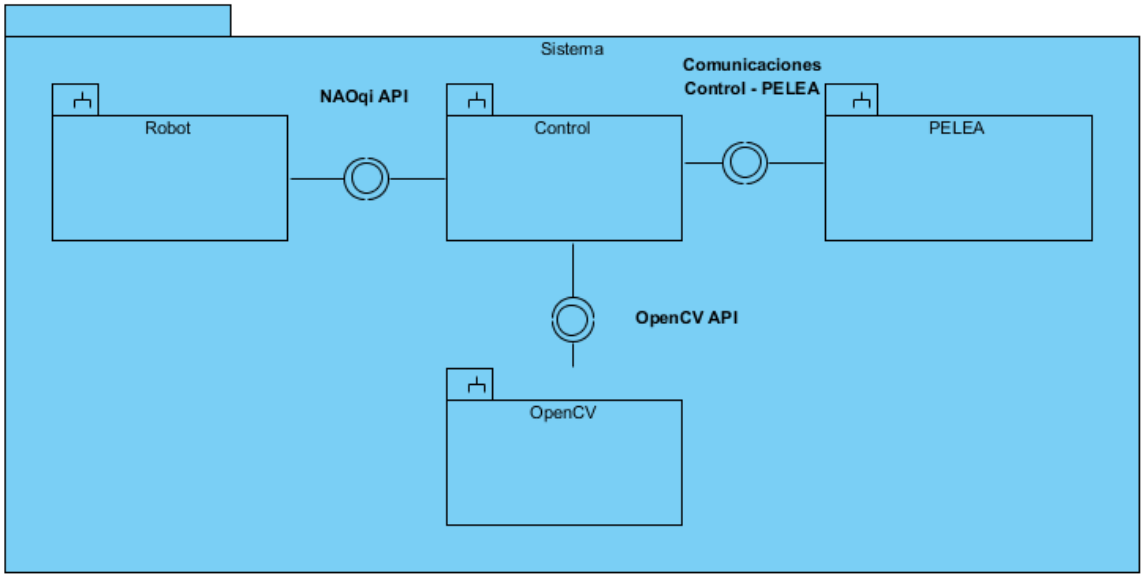


Figura 22 – Subsistemas

- Robot: Se corresponde con el conjunto de librerías que mandan órdenes al robot, que permitirán que éste sea capaz de moverse por el entorno y proveer de sensado al sistema mediante el hardware del robot (cámara, giroscopio, acelerómetro…). Además mediante las órdenes el robot podrá coger y depositar objetos. Se comunica exclusivamente con Control.
- Control: Se corresponde con el sistema de control híbrido desarrollado para este trabajo. Está encargado de controlar al robot, analizar y procesar la información de los sensores, traducir de alto nivel a bajo nivel las acciones del planificador y ofrecer una interfaz de

comunicación viable entre un humano y el robot. Se comunica con PELEA, el robot NAO y OpenCV, además gestiona las comunicaciones del sistema.

- PELEA: Se corresponde con la arquitectura de planificación automática PELEA. Está encargado de introducir las metas en el problema, resolver el problema descomponiéndolo en acciones y obtener la siguiente acción a realizar dado el estado del problema. Se comunica exclusivamente con Control.
- OpenCV: Se corresponde con la librería de OpenCV. Se encarga de proporcionar al sistema un conjunto de funciones para el procesamiento de imágenes. Se comunica exclusivamente con Control.

3.3.2 Descripción de componentes

En este apartado se describen de forma detallada los componentes del sistema, así como las interfaces que hacen posible la comunicación entre los diferentes subsistemas. En la Figura 23 se presenta un diagrama con la descomposición del sistema en componentes junto a sus interfaces.

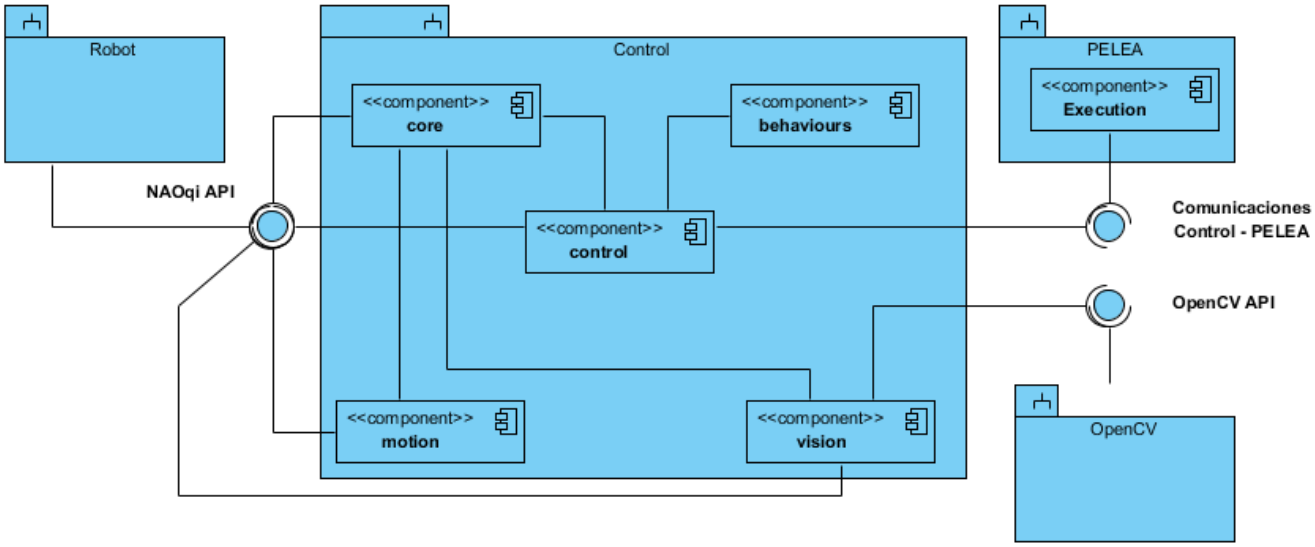


Figura 23 – Diagrama de componentes del sistema

3.3.2.1 Interfaces

Existen 3 interfaces que se encargan de conectar los 4 subsistemas:

- NAOqi API: Es una API de programación que permite acceder a los diferentes dispositivos (sensores, actuadores, memoria, etc) que integran el robot NAO. Ha sido desarrollada por la empresa Aldebaran Robotics.
- Comunicaciones Control - PELEA: Permite la comunicación entre el sistema desarrollado y la arquitectura PELEA. Debido a que ambos sistemas están desarrollados en lenguajes de programación diferentes se han utilizado comunicaciones basadas en TCP/IP.
- OpenCV API: Es una API de programación que permite acceder a diferentes técnicas de procesado de imagen. Ha sido desarrollada por la comunidad de Open Source Computer Vision.

3.3.2.2 Componente Vision

Este componente contiene todas las funciones de captura y procesamiento de imágenes. Mediante este componente han sido implementadas todas las funcionalidades relacionadas por los procesos de visión. Primero se realiza una descripción del proceso de selección de objetos (introducir metas) seguido del proceso de reconocimiento de objetos, ya que se tratan de los dos procesos principales de visión. Después se explica el mecanismo de obtención de imágenes mediante el robot seguido de una explicación en detalle de todas las técnicas utilizadas para procesar las imágenes, seguido del proceso para guardar un objeto, reconocer un contenedor y calcular la distancia al objeto.

3.3.2.2.1 Selección de objetos

El almacenamiento de objetos en el sistema u la introducción de metas, requiere interacción con el usuario, que interactuará con el robot para llevar a cabo ésta labor. Una vez que el sistema inicia la “fase” de almacenamiento de objetos, el robot indicará al usuario mediante sus altavoces, que tiene que introducir las metas en el sistema, para ello el usuario tendrá que mostrarle uno o varios objetos al robot.

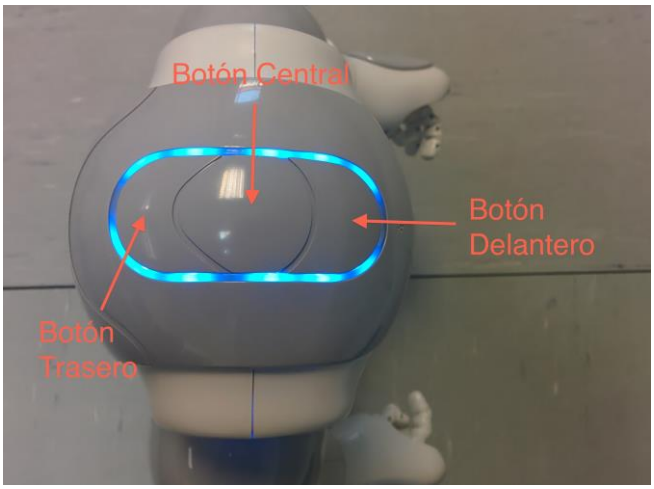


Figura 24 – Botones táctiles del robot situados en la cabeza

Para hacer posible la interacción entre el robot y el usuario, la persona podrá seleccionar y navegar por los "menús" (Identificando, Preguntando y Confirmando) de la interfaz que ofrece el sistema utilizando los botones táctiles situados en la cabeza del robot (Figura 24 – Botones táctiles del robot situados en la cabeza). Además para facilitar el reconocimiento de los objetos, se mostrará en una pantalla la cámara del robot en modo vídeo (Figura 25(a) y Figura 25(b)), para poder ayudar al usuario a colocar los objetos delante de la cámara del robot.

(a) (b)

Figura 25 – Cámara del robot en modo vídeo con área de captura 1 (a) Cámara del robot en modo vídeo con área de captura 2(b)

El funcionamiento de la fase de almacenamiento de objetos se consta de 3 menús o estados posibles que se describen a continuación:

- Preguntando: Estado inicial. El robot mediante sus altavoces, indica al usuario que tiene que almacenar objetos; además explica las posibles acciones a realizar: Botón Delantero – Comenzar a identificar objeto (cambia al estado Identificando), Botón Central – Repetir éstas instrucciones, Botón Trasero – Comenzar el sistema de planificación y resolver el problema (Es necesario haber introducido algún objeto, en caso negativo, el robot se lo indicará al usuario y volverá a repetir las acciones posibles).
- Identificando: Al comenzar este estado, el robot mediante sus altavoces, indica al usuario cuantos objetos se han almacenado hasta el momento; además explica las posibles acciones a realizar: Botón Delantero – Capturar y procesar imagen (cambia al estado Confirmando), Botón Central – Aumenta el área de reconocimiento (Comparación entre Figuras 25(a) y 25(b)), Botón Trasero – Informa al usuario del número de objetos almacenados hasta el momento y permite terminar de identificar objetos (cambia al estado Confirmando).
- Confirmando: Se utiliza para dar la posibilidad al usuario de confirmar el almacenamiento del objeto capturado o terminar de identificar objetos. Al entrar en este estado, el robot mediante sus altavoces explica al usuario las posibles acciones a realizar: Botón Delantero – Confirmar acción, Botón Central – Repetir éstas instrucciones, Botón Trasero – Negar acción; donde acción puede ser guardar objeto o terminar de identificar objetos. Si la acción es terminar de identificar objetos y se confirma, se transitará al estado preguntando; en caso negativo se transitará al estado identificando. Si la acción es almacenar objeto y se confirma, se almacenará el objeto y en caso negativo no se almacenará; para ambas posibilidades se transitará al estado identificando.

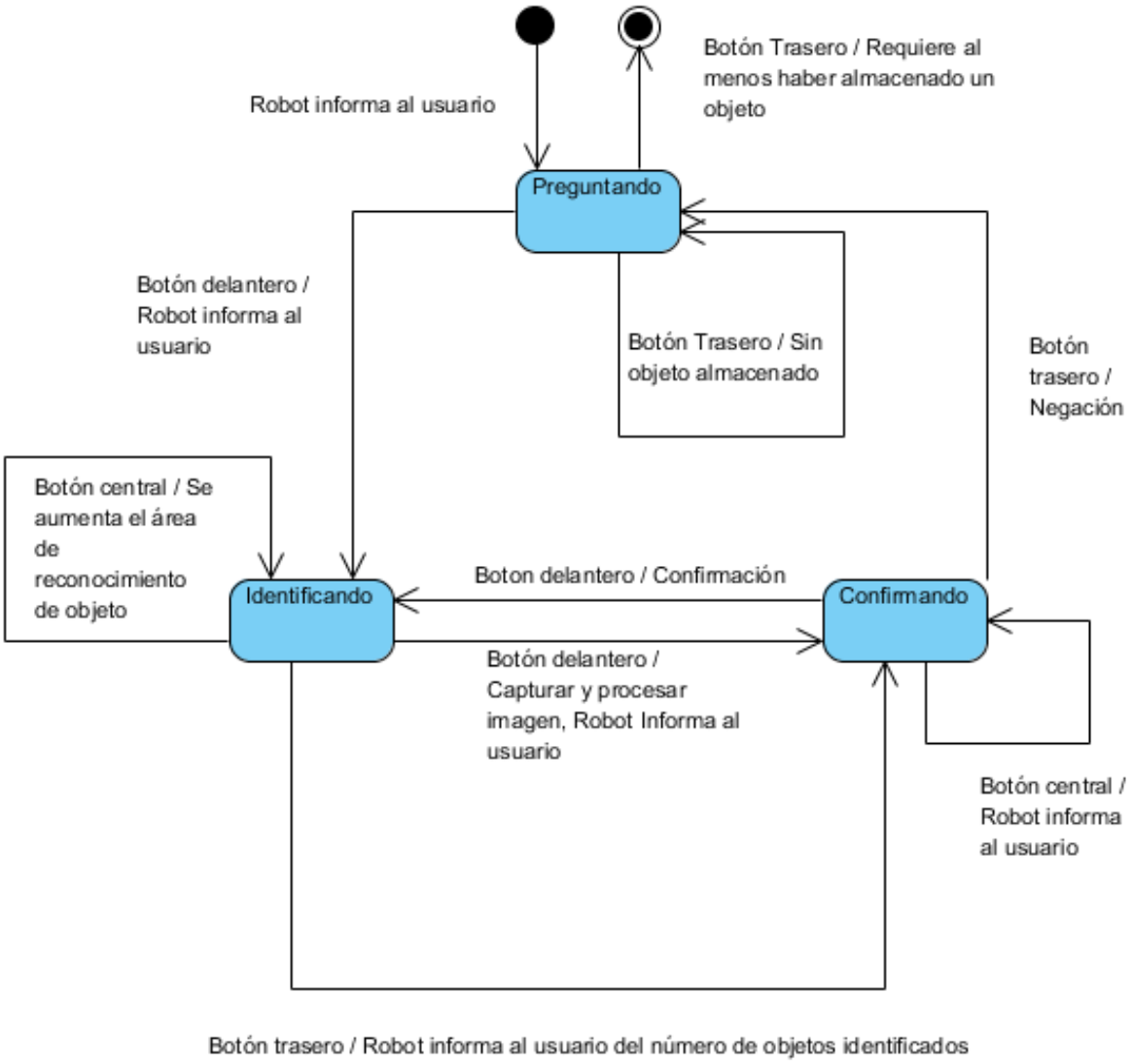


Figura 26 – Diagrama de flujo de la tarea de selección de objetos

En la figura 26 se presenta el diagrama de flujo de la tarea de selección de objetos, donde se pueden ver los diferentes estados y sus transiciones explicados en el apartado anterior. En la demostración del problema 1B se puede ver el funcionamiento real de este proceso.

3.3.2.2.2 Reconocer objeto

Para poder recoger lun objeto, el sistema debe ser capaz de reconocerlo mediante visión. En la Figura 27 se muestra el flujo de reconocimiento, el orden de ejecución y el posible resultado de esta tarea.

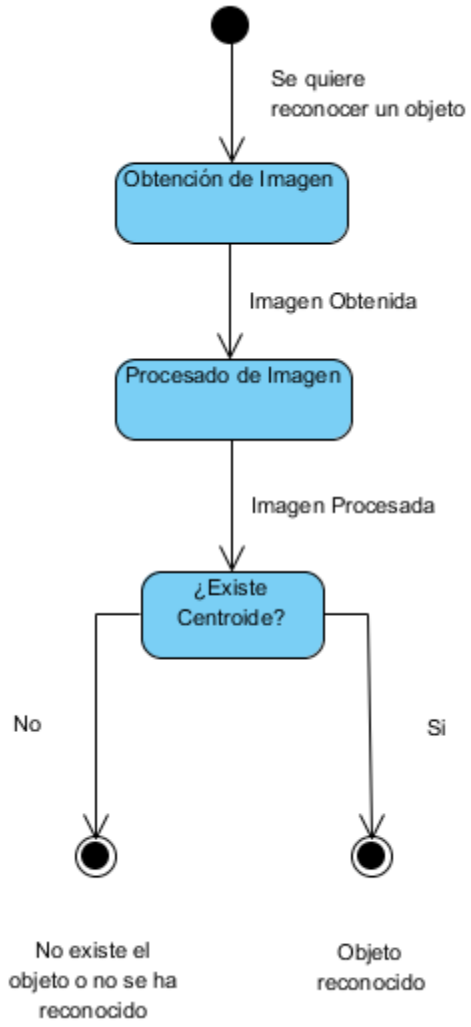


Figura 27 – Diagrama de flujo de la tarea de reconocer objeto

Para reconocer un objeto, es necesario:

1. Una imagen capturada por una de las cámaras del robot (apartado 3.3.2.2.3)
2. El color identificativo del objeto (apartado 3.3.2.2) en espacio HSV.
3. Procesar la imagen. Se aplican los pasos 1-7 del apartado 3.3.2.2.2. En el paso 5 se utiliza el color identificativo del objeto del punto 2.
4. Buscar el objeto. Si se ha hallado un centroide, significa que existe el objeto y las coordenadas del centroide corresponderán aproximadamente con el centro del objeto que se quiere reconocer. En la Figura 28 se puede ver un objeto reconocido mediante visión.

Figura 28 – Objeto reconocido

3.3.2.2.3 Obtención de imágenes

Las imágenes se obtienen directamente de una de las cámaras del robot utilizando el módulo ALVideoDevice. Éstas imágenes se obtienen en el modelo de color RGB de 8 bits en una matriz de píxeles de 320 x 240. En la Figura 27 se puede ver una imagen capturada por la cámara inferior del robot.

Figura 29 – Imagen capturada por una de las cámaras del robot NAO

En las fases iniciales de desarrollo y para poder probar la funcionalidad de detección de objetos, se utilizaron diversas cámaras web, posteriormente, para probar con la configuración específica de la cámara del robot, se usó el simulador para probar la detección de objetos. Una vez que se pudieron detectar los objetos en el simulador, se pasó a utilizar el robot real para terminar de

ajustar los parámetros de identificación, ya que la iluminación generada por el simulador y la del mundo real son diferentes.

3.3.2.2.4 Procesado de la imagen

Para el procesado de la imagen se ha usado la librería de OpenCV para Python. En la Figura 30 se presenta el diagrama de flujo del procesamiento de una imagen con las transformaciones que se enumeran a continuación:

1. Imagen obtenida en modelo de color RGB de 8 bits (Ilustración 1(a)).
2. Cambiar el modelo de color a BGR (Ilustración 2(a)).
3. Aplicar un difuminado a la imagen (Ilustración 2(b)).
4. Cambiar el modelo de color a HSV (Ilustración 3(a)).
5. Dado un rango de color, cambiar la imagen a blanco y negro, sustituyendo los valore del rango de color por el blanco y el resto por negro (Ilustración 3(b)).
6. Aplicar un suavizado, erosionado y dilatado a la imagen en blanco y negro(Ilustración 4(a)).
7. Calcular el centroide de los píxeles en blanco (Ilustración 4(b)).

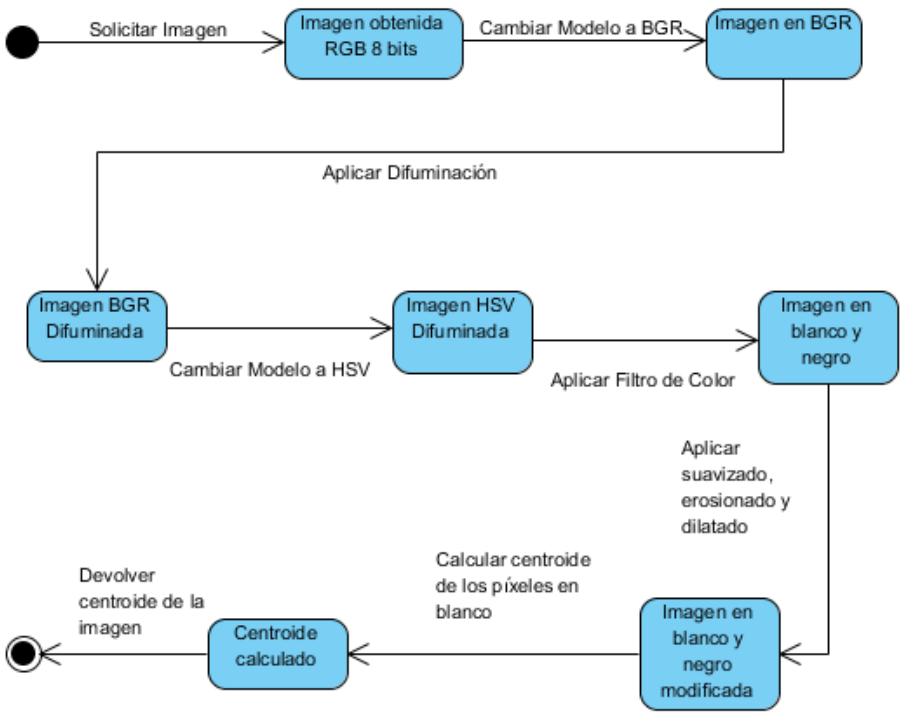


Figura 30 – Diagrama de flujo del procesamiento de una imagen

Es necesario cambiar el modelo de color a BGR (Figura32(a)) ya que OpenCV por defecto usa ese modelo y la imagen original viene en formato RGB, se puede apreciar la diferencia entre los dos formatos en las Figuras 31(a) y 31(b). Se aplica un difuminado a la imagen (Figura 32(b)) para que el color sea más uniforme y reducir la diferencia de color entre píxeles cercanos. El cambio de modelo a HSV (Figura 33(a)) se realiza para poder filtrar a través del tono (Hue) (Figura 33(b)), ya que de esta manera, identificar los colores mucho más sencillo. Los suavizados, erosionados y dilatados (Figura 34(a)) se realizan para facilitar el cálculo del centroide. Para descartar posibles falsos positivos en la detección del centroide, se ha puesto un límite inferior al área total representada por puntos blancos en la imagen, si el área es inferior al límite, se entiende que la agrupación del color no es lo suficientemente grande como para poder interpretarlo como un objeto y se descarta el centroide en este procesado de imagen; el límite se ha establecido por medio de prueba y error, encontrando el valor que mejor se ha ajustado para detectar el objeto e ignorar en caso de que no estuviese presente. Finalmente, en caso de superar el límite anterior, se obtiene el centroide indicando la posición del objeto (Figura34(b)).

A continuación se muestran los pasos en el procesado de imagen con capturas reales de cada apartado.

(a) (b)

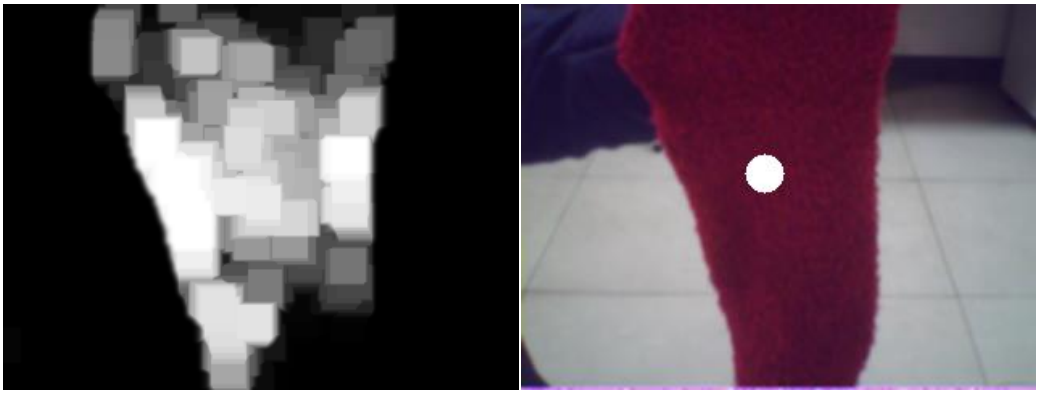
Figura 31 – Imagen original (a) Imagen de OpenCV (b)

(a) (b)

Figura 32 – Imagen OpenCV cambiada a BGR (a) Imagen difuminada (b)

(a) (b)

Figura 33 – Imagen en formato HSV (a) Imagen en blanco y negro (b)



(a) (b)

Figura 34 – Imagen con suavizado, dilatado y erosionado (a) Centroide encontrado sobre la imagen original (b)

3.3.2.2.5 Guardar objeto

El sistema necesita almacenar objetos para poder identificarlos y recogerlos posteriormente. El diseño de esta tarea se ha realizado en función de los objetos que el robot podría ser capaz de recoger. En la Figura 35 se muestra la mano del robot NAO sujetando un pincel.

Figura 35 – Robot Nao sosteniendo un pincel envuelto

Como se puede observar en la Figura 35, el robot tiene una serie de limitaciones físicas debido a la estructura de su mano. Debido a esto, el objeto tendría que cumplir las siguientes características:

- Ser pequeño, ya que tiene que poder caber en la mano del robot.
- Ser ligeramente compresible, ya que permitirá al robot comprimirlo sin estropear el objeto.
- Ser ligeramente rugoso, para evitar el deslizamiento dentro de la mano del robot.
- Tener sólo un color o al menos un color predominante. Ésta característica no está definida por las limitaciones físicas del robot, pero facilitará la identificación y reconocimiento del objeto.

Con las características anteriores, se seleccionó como objeto a recoger un calcetín (aunque el sistema es capaz de recoger cualquier objeto con características similares). Por la naturaleza del objeto a recoger, apareció una nueva condición:

- El objeto puede cambiar de forma, es decir, cuando se muestre el objeto al robot para almacenarlo no tiene por qué tener la misma forma que cuando el robot tenga que recogerlo.

Debido a esta condición, inicialmente se descartó el reconocimiento de objetos por su forma, pero se desarrolló un mecanismo de detección mediante contornos que se detalla al final de este apartado. Finalmente, se decidió utilizar identificación y reconocimiento de objetos mediante color.

Para poder almacenar los objetos mediante su color, se toma una imagen con el robot, se aplican los pasos 1-4 del apartado 3.3.2.2.4, después, se calcula el color predominante dentro de una circunferencia, con centro en el centro de la imagen y de radio el establecido por el usuario mediante los botones del robot. El color predominante viene dado por un rango entre 0-180, ya que es el rango que utiliza OpenCV para el tono (Hue) con el modelo de color HSV. Éste rango de color será el identificador del color que servirá para poder reconocerlo posteriormente. De esta manera se permite almacenar cualquier objeto con características similares a un calcetín, cumpliendo las restricciones descritas en éste apartado. En la Figura 36 se muestra cómo se guarda un objeto mediante visión.

Figura 36 – Objeto guardado mediante visón

Se probó la detección mediante contornos, pero se descartó debido a que tenía mayor índice de error que la detección por color, ya que la detección por contornos requiere de una figura geométrica para que funcione adecuadamente, pero el objeto en cuestión no siempre tiene esta forma, por lo que la detección por contornos fallaba con frecuencia.

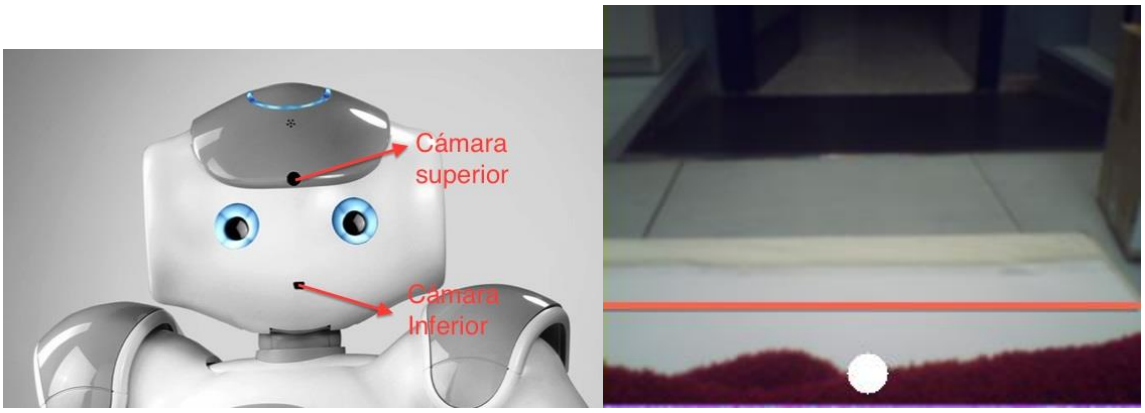
3.3.2.2.6 Reconocer contenedor

Para conocer la ubicación del contenedor donde el robot deberá depositar los objetos, es necesario reconocerlo, para ello el contenedor también tendrá un identificador visual para poder detectarlo mediante visión. El mecanismo de detección del contenedor es idéntico al de reconocimiento de un objeto. En la Figura 37 se muestra como el sistema es capaz de reconocer el contenedor mediante su identificador de color (verde).

Figura 37 – Contenedor reconocido

3.3.2.2.7 Calcular distancia al objeto

Para que el robot pueda aproximarse al objeto y colocarse a la distancia adecuada, el sistema necesita de un mecanismo para medir la distancia o profundidad. Para medir la distancia del robot al objeto se ha usado el mecanismo para reconocer objetos (3.3.2.2.2) en combinación con las dos cámaras del robot (Figura 38(a)) y la posición del cuello del robot. Existen tres posibles estados y cada uno determina una distancia. Cada estado viene descrito por la cámara que utiliza y la posición del cuello en negrita. Los estados son los siguientes:



(a) (b)

Figura 38 – Cámaras del robot (a) Límite para cambiar de cámara del robot(b)

- **Cámara superior del robot** (Figura 38(a)): Se interpreta que el objeto todavía está lejos, al menos a más de media celda de distancia y el robot avanzará de manera normal. Si el centroide obtenido al reconocer el objeto, se encuentra por debajo de un límite de altura de la imagen (se ha usado 3.5/5 del alto de la imagen, ver Figura 38(b)), se entiende que el robot ya está cerca del objeto y si avanza otra vez de la misma manera, perderá de vista el objeto. En este punto, se pasa a usar la cámara inferior del robot y se inclina la cabeza del robot hacia atrás para mantener un buen rango de visión.
- **Cámara inferior del robot y la cabeza está inclinada hacia atrás** (Figura 39(a)): Se interpreta que el robot está cerca del objeto, al menos a menos de media celda de distancia y el robot avanzará dando pequeños pasos para evitar perder de vista el objeto o avanzar más de lo necesario. Si el centroide obtenido al reconocer el objeto, se encuentra por debajo de un límite de altura de la imagen (se ha usado el mismo que para el apartado 1), se entiende que el robot ya está muy cerca del objeto y si avanza otra vez de la misma manera, perderá de vista el objeto. En este punto, la posición de la cabeza del robot se deja en su posición por defecto para seguir manteniendo a la vista el objeto en el tramo final de aproximación.
- **Cámara inferior del robot y la cabeza está en su posición normal** (Figura 39(b)): Se interpreta que el robot está muy cerca del objeto y en pocos pasos llegará a la posición adecuada. El robot avanzará de esta manera hasta que recoja el objeto.

(a) (b)

Figura 39 – Robot usando cámara 2 con cabeza inclinada (a) Robot usando cámara 2 con cabeza en posición normal (b)

3.3.2.3 Componente Behaviours

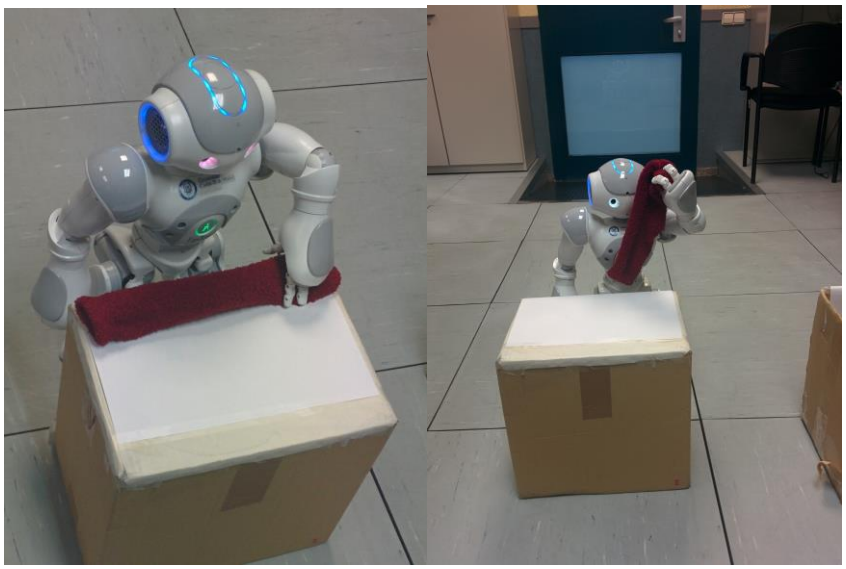
Se trata del componente dentro del sistema Control y contiene todas las primitivas de movimiento del robot. Una primitiva es una función atómica que se corresponde con una operación básica de las permitidas por el robot. Como por ejemplo abrir una mano. De forma que un comportamiento está compuesto por un conjunto de primitivas que se corresponden con una secuencia de acciones en las articulaciones del robot, cuya ejecución secuencial producirá un movimiento complejo. Se utilizarán solamente dos primitivas que se van a describir a continuación.

3.3.2.3.1 Coger objeto

Una vez que el robot se ha posicionado adecuadamente, esta primitiva permite al robot coger el objeto. La primitiva está compuesta por la siguiente secuencia de acciones:

1. Abrir la mano.
2. Posicionar la mano encima del objeto y bajarla hasta que pueda agarrarlo (Figura 40A).
3. Cerrar la mano 4 veces.
4. Levantar ligeramente el brazo y llevarlo hacia el lateral para evitar que roce con la superficie donde se encuentra el objeto (Figura 40(b)).
5. Cerrar la mano 2 veces.
6. Situar el brazo en posición de caminar.

7. Cerrar la mano 2 veces.



(a) (b)

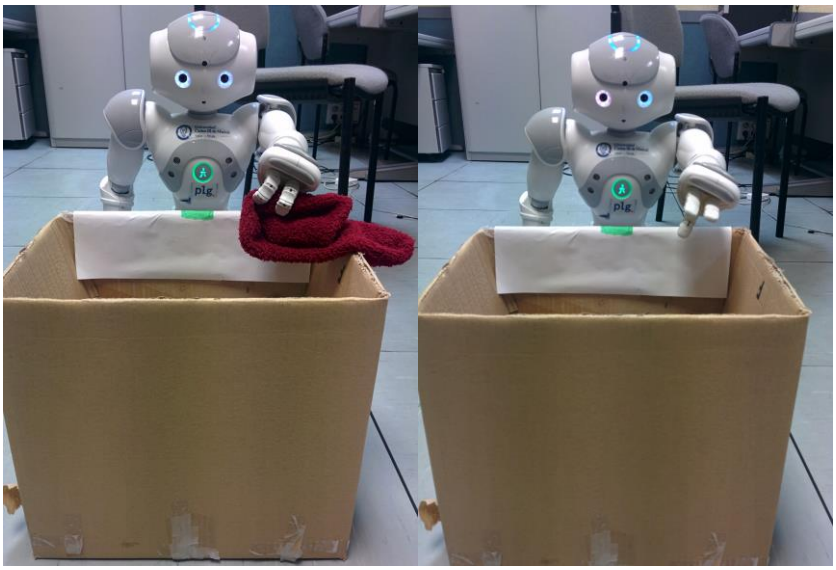
Figura 40 – NAO intentando coger el objeto (a) Robot Nao sosteniendo el objeto (b)

El motivo por el cual hay que cerrar la mano del robot es debido a que no puede hacer mucha fuerza con la mano y es fácil que se le pueda caer el objeto mientras se realizan otros movimientos.

3.3.2.3.2 Soltar objeto

Una vez que el robot se ha posicionado adecuadamente, esta primitiva permite al robot depositar el objeto. La primitiva está compuesta la siguiente secuencia de acciones:

- Situar la mano encima del contenedor y girar el brazo de tal manera que la palma de la mano del robot esté boca abajo (Figura 41(a)).
- Abrir la mano (Figura 41(b)).
- Situar la mano en posición de caminar.



(a) (b)

Figura 41 – NAO intentando depositar el objeto en el contenedor (a) Robot Nao ha depositado el objeto en el contenedor (b)

3.3.2.3 Componente Motion

Se trata del componente dentro del sistema Control y contiene todas las funciones de movimiento y aproximación al objeto.

3.3.2.3.1 Calcular error en la trayectoria

El robot se desvía de su trayectoria al indicarle que camine hacia delante en línea recta, esto se debe principalmente a la forma de caminar y al tambaleo generado por el movimiento. Normalmente se suele desviar hacia la izquierda, este error se corrige en parte haciéndole caminar hacia delante con una ligera desviación a la derecha, pero aun así hay veces en las que se desvía. Para corregir este error, se ha utilizado el proceso de reconocimiento de objetos (3.3.2.2.2). Utilizar ésta técnica para corregir el error en la trayectoria hace que tenga gran precisión cuando el robot pueda ver al objeto, es decir, si el robot no está orientado hacia el objeto, no se recolocará. En ocasiones (muy pocas) el robot se desvía tanto de su trayectoria que pierde el contacto de visión con el objeto y ya no es posible aplicar ningún tipo de corrección.

Para calcular el error, se obtiene la posición del centroide, y se compara con el centro de la imagen. Si la diferencia supera un límite (corresponde con el 10% del ancho de la imagen desde el centro de la imagen), el robot debe reposicionarse, orientándose hacia el objeto en el sentido que

sea necesario. Los pasos anteriores se repiten hasta que la diferencia de la posición del centroide respecto al centro de la imagen sea inferior al límite. Una vez que se ha corregido el error, se puede continuar con la ejecución del sistema. En la Figura 42 se puede observar que el robot tendrá que girar a la izquierda para orientarse adecuadamente hacia el objeto, el objeto reconocido aparece con el círculo blanco, el centro de la imagen con la línea continua y el límite máximo para no tener que reposicionarse viene determinado por las líneas discontinuas. Si el objeto no se encuentra entre el segmento de línea discontinua, centro de la imagen y línea discontinua, deberá recolocarse.

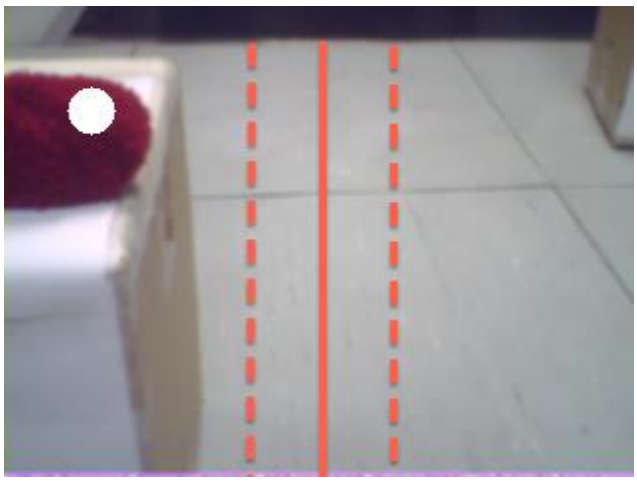


Figura 42 – Diferencia de posición del objeto respecto al robot

3.3.2.3.2 Identificar posición para recoger objeto

Para recoger el objeto, se utiliza una secuencia de posiciones (Primitivas 3.3.2.3 Componente behaviours) del brazo del robot que permite recoger el objeto siempre y cuando el robot y el objeto se encuentren en las posiciones adecuadas. El objeto no variará de posición, por tanto, el robot es el que se tiene que ubicar en función del lugar en el que se encuentre el objeto. Para aproximarse al objeto, se utiliza el proceso de reconocimiento de objetos (3.3.2.2.2), el proceso de calcular la distancia al objeto (3.3.2.2.7) y el proceso de calcular el error en la trayectoria (3.3.2.3.1). Una vez que el robot se encuentra suficientemente cerca del objeto, tal y como se indica en los puntos 2 y 3 del apartado 3.3.2.2.7, se comprueba en cada iteración de avance la posición del objeto. La posición del objeto se obtiene mediante la posición de su centroide y para determinar si el robot puede coger o no el objeto, se secciona la imagen en 8 áreas en forma de

rectángulo. Dependiendo del área donde esté el centroide, se le ordenará al robot que se mueva o que recoja el objeto. Las 8 áreas y sus acciones son las siguientes:

- A1: El robot dará un paso hacia delante y otro lateral a la derecha.
- A2: El robot dará un paso hacia delante.
- A3: El robot dará un paso hacia delante.
- A4: El robot dará un paso hacia delate y otro lateral a la izquierda.
- A5: El robot dará un paso lateral a la derecha.
- A6: El robot recogerá el objeto con la mano izquierda.
- A7: El robot recogerá el objeto con la mano derecha.
- A8: El robot dará un paso lateral a la izquierda.

En la Figura 43 se puede ver la división de la imagen en las 8 áreas mencionadas, también se puede apreciar que el objeto se encuentra en el área 6.

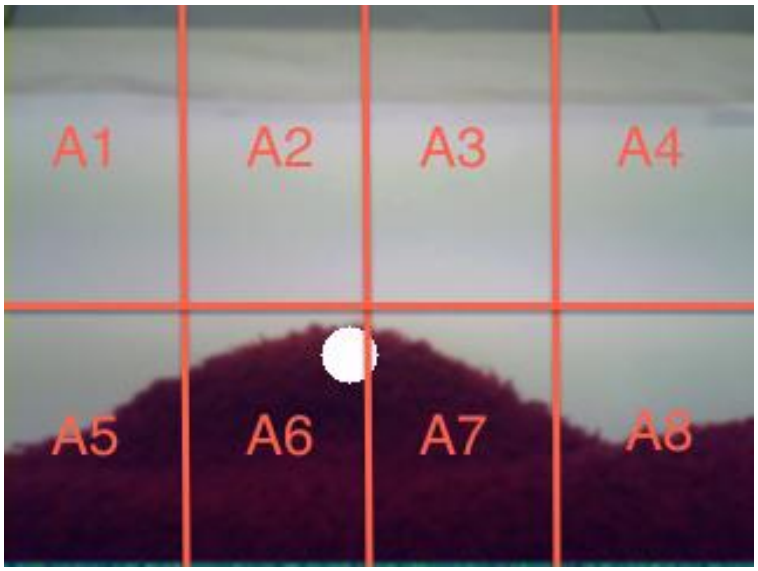


Figura 43 – Imagen seccionada en 8 áreas

3.3.2.5 Componente Core

Se trata del componente principal del sistema de Control que contiene la funcionalidad Motion y Vision. Ofrece una capa de abstracción entre los componentes Motion y Vision, para facilitar al componente Control la gestión de la funcionalidad y la información de ambos.

3.3.2.5 Componente Control

Este componente se encarga de coordinar los otros componentes del sistema y establecer el bucle de control y comunicarse con el resto de subsistemas. Además también adapta las acciones del planificador en movimientos del robot (traducción alto nivel – bajo nivel), proporciona una interfaz de comunicación entre el usuario y el robot y lleva a cabo las correcciones del robot usando sus sensores.

3.3.2.5.1 Corrección del giro

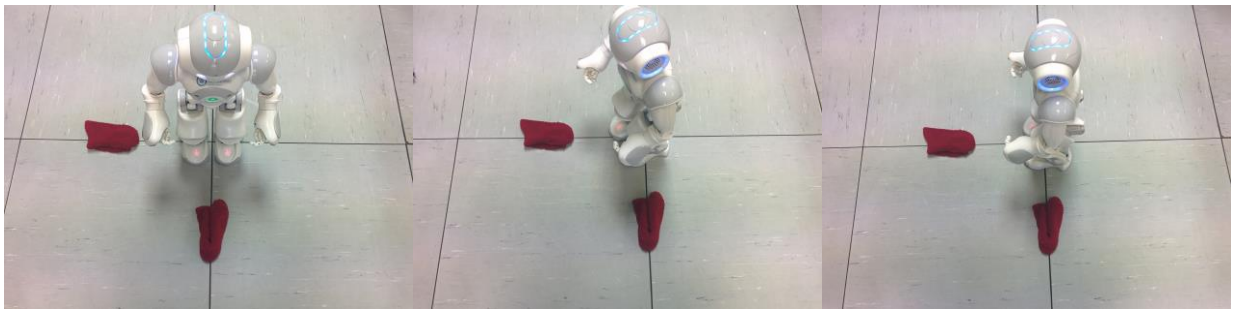
Cuando al robot se le da la orden de caminar o girar, la mayoría de las veces el robot lo realiza con error. Si el robot tiene que realizar un giro, es posible calcular los grados que el robot ha girado sobre sí mismo mediante la medición del los valores del giroscopio. Un giroscopio es un dispositivo para medir o mantener la orientación basado en el principio angular.

Los valores del giroscopio cuentan con el error de la propia medida del dispositivo (tal y como se detalla en las especificaciones de hardware del robot) y de la obtención de los valores mediante la API del robot. El error debido a la obtención de los valores del robot se debe a que la tasa de refresco de estos valores es de 10ms, pero lo más rápido que se ha conseguido acceder a los valores de memoria es de 50ms, por lo tanto, se pierde la información de 4 de cada 5 valores; esto se debe a las comunicaciones entre el robot y el sistema de control, donde no es posible acceder a la información del robot a tanta velocidad ya que el sistema de control se encuentra fuera del robot y las comunicaciones tienen un coste de tiempo. Aunque no se consiga acceder a toda la información, se puede obtener un valor bastante cercano al giro realizado por el robot (el error medio se encuentra alrededor de 5º). Los valores que se obtienen del robot corresponden con la velocidad angular (ω) dada en radianes · segundo^{-1}. Para estimar el ángulo de giro ($\Delta\theta$) se utiliza la siguiente fórmula:

Donde t, es el tiempo, y k es una constante k=1.4 para aproximar el ángulo de giro debido a la desincronización en la toma de los valores del robot.

$$\Delta\theta = \frac{\sum_{i=0}^{n} \omega}{n} \; i_{max} \; t \; k$$

En las imágenes presentadas en la Figura 44, se puede apreciar un giro de 90º del robot que no se ha ejecutado adecuadamente, y como con el valor del giroscopio se corrige ese error.



(a) (b) (c)

Figura 44 – Posición antes del giro (a) Giro de 90º fallido (b) Corrección del giro por los valores del giroscopio(c)

Debido al tambaleo producido por el robot al desplazarse y la desincronización en la toma de los valores del robot, este método sólo se puede aplicar cuando el robot se mueve en el eje Z (girar derecha o izquierda), ya que en los desplazamientos en el eje X e Y los valores del robot tienen mucho ruido y es una cantidad insuficiente como para poder dar resultados precisos.

3.3.2.5.2 Traducción alto nivel a bajo nivel

Para resolver los escenarios propuestos para el sistema, se utiliza planificación automática. Mediante la planificación es posible saber si el problema propuesto se puede solucionar y en caso afirmativo, descomponer la solución del problema en acciones. Estas acciones, se enviarán a Control y este se encargará de descomponerlo (traducción de alto nivel a bajo nivel) en primitivas u otros comandos que el robot será capaz de interpretar y realizar. De esta manera, se pueden efectuar las acciones del planificador mediante el robot. Aunque se utiliza un sistema de planificación automática, no todo el sistema funciona de manera deliberativa. La parte deliberativa del sistema corresponde con la descomposición de la solución del problema en acciones, pero algunas acciones se complementan con una parte reactiva usando la visión artificial.

El planificador a utilizar es Metric-FF[23] y el lenguaje de planificación automática es PDDL, teniendo como único requisito de semántica para el dominio :typing. El dominio que será capaz de resolver los problemas aptos para el robot será sencillo, con tan solo 4 acciones que se explicarán en detalle más adelante. Se trata de un número pequeño de acciones ya que se ha mantenido un alto nivel de abstracción respecto a las funciones realizadas por el robot; estas 4 acciones podrían haberse descompuesto en otras más pequeñas pero por razones de sencillez se ha decidido este

diseño. Además el dominio permite depositar objetos en orden. En la Figura 45 se presenta la definición de tipos y predicados del dominio PDDL que ha sido desarrollado para este trabajo.

```
(define (domain NAO)
        (:requirements :typing)
        (:types orientation cell color order)
        (:predicates
                (linked ?c1 - cell ?c2 - cell ?o - orientation)
                (robot ?c1 - cell ?o1 - orientation)
                (blocked ?c1 - cell)
                (item ?color - color ?c1 - cell ?o - orientation)
                (container ?color - color ?c1 - cell ?o - orientation)
                (hold ?color - color)
                (busy)
                (current-goal ?goal - order)
                (delivered ?color - color)
                (position ?color - color ?order - order)
                (next ?before - order ?after - order)
                (end-goal ?goal - order)
                (finished)
        )
```

Figura 45 – Objetos y predicados del dominio de planificación automática

Los ficheros de problemas a resolver por el dominio, deberán contener:

- Un mapa del tamaño del problema representado mediante celdas enlazadas por doble sentido.
- 4 Objetos con los puntos cardinales para mantener la orientación entre celdas y las posibles transiciones entre ellas.
- La ubicación inicial del robot junto con su orientación fijada como norte. Es necesario que comience con esta orientación para mantener la sincronización con el Control.

- La posición del objeto u objetos y ubicaciones de entrega en la celda o celdas correspondientes.

En caso de querer extender la funcionalidad, bastará con modificar el dominio de planificación y adaptar en caso de que se hayan creado, la acción o acciones añadidas al dominio; también hay que actualizar el módulo Execution de PELEA para que contemple los efectos realizados por las nuevas acciones.

Las cuatro acciones que contempla el dominio de planificación automática son Move, Turn, Catch, y Drop. A continuación se presenta una descripción de cada una de ellas junto con su representación en alto (PDDL) y bajo nivel.

1. Move

Permite al robot moverse entre celdas, sólo podrá desplazarse mediante celdas enlazadas (adyacentes y sin movimientos oblicuos). En la Figura 46 se presenta la representación de la acción en código PDDL.

```
(:action move
        :parameters (?c1 - cell ?c2 - cell ?o - orientation)
        :precondition (and (robot ?c1 ?o) (linked ?c1 ?c2 ?o) (not (blocked ?c2)) )
        :effect (and (robot ?c2 ?o) (not (robot ?c1 ?o)) )
)
```

Figura 46 – Acción “move” en el dominio de planificación automática

La traducción de esta acción está formado por:

- Mover al robot en línea recta una cuadrícula.
- Corregir el movimiento error en el movimiento anterior.

Nota: Se realiza la corrección porque el robot tiende desviarse a la derecha. La corrección es la siguiente: un giro a la izquierda de 20º, cuatro pasos laterales a la izquierda y otro giro de 20º.

2. Turn

Permite al robot orientarse (girar) hacia cualquiera de los puntos cardinales. En la Figura 47 se presenta la representación de la acción en código PDDL.

```
(:action turn
        :parameters (?c1 - cell ?o1 - orientation ?o2 - orientation)
        :precondition (and (robot ?c1 ?o1))
        :effect (and (robot ?c1 ?o2) (not (robot ?c1 ?o1)) )
)
```

Figura 47 – Acción "turn" en el dominio de planificación automática

La traducción de esta acción está formado por:

- Comenzar a grabar los valores del giroscopio del robot.
- Girar al robot 90º derecha o izquierda según sea necesario.
- Parar de grabar los valores del giroscopio, calcular el error producido y corregirlo.
- En caso de que sea necesario realizar otro giro (giros de 180º), se repetirán los puntos anteriores.

3. Catch

Mediante esta acción el robot recoge un objeto situado en una celda. En la Figura 48 se presenta la representación de la acción en código PDDL.

```
(:action catch
        :parameters (?c2 - cell ?o - orientation ?color - color ?before - order)
        :precondition (and (not (busy)) (item ?color ?c2 ?o) (robot ?c2 ?o)
                (not (delivered ?color)) (current-goal ?before) (position ?color ?before))
        :effect (and
                (hold ?color) (busy)
                (not (item ?color ?c2 ?o)))
)
```

Figura 48 – Acción "catch" en el dominio de planificación automática

La traducción de esta acción está formado por:

- Tomar una imagen y su centroide.
- Acercarse al objeto. Se explica este funcionamiento en el apartado 3.3.2.3.1 Calcular distancia al objeto.
- Calcular error mediante visión y corregir la orientación. Se explica este funcionamiento en el apartado 3.3.2.3.2 Calcular error en la trayectoria.
- Repetir los puntos 2, 3 y 4 hasta que el robot esté posicionado adecuadamente para coger el objeto.
- Cargar la primitiva 3.3.2.3.1 Coger objeto.
- Hacer que el robot de dos pasos hacia atrás y establecer como objeto el identificador de color del contenedor donde el robot deberá depositar.

4. Drop

Mediante esta acción el robot suelta un objeto previamente recogido. En la Figura 49 se presenta la representación de la acción en código PDDL.

```
(:action drop
        :parameters (?c2 - cell ?o - orientation ?color - color ?before - order ?after - order)
        :precondition (and (hold ?color) (container ?color ?c2 ?o) (robot ?c2 ?o)
                (not (delivered ?color)) (current-goal ?before)
                (next ?before ?after) (position ?color ?before))
        :effect (and
                (item ?color ?c2 ?o) (delivered ?color) (current-goal ?after)
                (not (hold ?color)) (not (busy)) (not (current-goal ?before)))
)
```

Figura 49 – Acción "drop" en el dominio de planificación automática

La traducción de esta acción está formado por:

- Tomar una imagen y su centroide.
- Acercarse al contenedor donde el robot deberá depositar el objeto. Se explica este funcionamiento en el apartado 3.3.2.3.1 Calcular distancia al objeto.
- Calcular error mediante visión y corregir la orientación. Se explica este funcionamiento en el apartado 3.3.2.3.2 Calcular error en la trayectoria.
- Repetir los puntos 2, 3 y 4 hasta que el robot esté posicionado adecuadamente para depositar el objeto.
- Cargar la primitiva 3.3.2.3.2 Soltar objeto.
- Hacer que el robot de dos pasos hacia atrás y continuar con la siguiente meta.

3.3.2.5.2 Comunicación con robot NAO

La comunicación entre el robot y el subsistema de Control se realiza a través de la API NAOqi, utilizando como lenguaje de programación Python. La API, ofrece diversos módulos a los cuales se puede conectar mediante comunicaciones TCP/IP.

3.3.2.5.1 Comunicación con PELEA

La comunicación entre PELEA y el Control se realiza mediante un comunicaciones TCP/IP. Esto se debe a que PELEA está en JAVA y el Control está en Python. El protocolo de comunicación usado consiste en un conjunto de mensajes de confirmación o negación, indicando que el mensaje ha sido recibido y que el contenido del mensaje se ha procesado adecuadamente o ha ocurrido algún error. De la comunicación entre ambos subsistemas sólo cabe destacar que el Control inicia PELEA mediante un script para lanzar todos los módulos necesarios para que funcione correctamente.

3.3.2.7 Componente Execution

Se trata del Módulo de Execution de la arquitectura PELEA adaptado para este trabajo. PELEA permite en su módulo Execution implementar la funcionalidad necesaria para comunicarse con cualquier robot, enviarle las acciones obtenidas por el planificador y actualizar el estado del problema en función de la información recibida por el robot. También es posible extender la funcionalidad adicional del resto de los módulos de PELEA, pero para este proyecto, sólo ha sido necesario adaptar el módulo Execution. Las tareas desarrolladas en el módulo de Execution son las siguientes:

- Introducir metas de forma dinámica: Debido a que las diferentes metas dependen del problema y del usuario, fue necesario implementar un mecanismo para recibir las metas del problema e introducirlas en el estado del problema.
- Protocolo de comunicación entre Control y PELEA: PELEA establece un determinado protocolo de comunicaciones, debido a esto ha sido necesario implementar un pequeño protocolo de comunicación para hacer posible el intercambio de mensajes.
- Actualizar estado del problema en función de la respuesta recibida del Control: Cuando PELEA envía una acción del planificador, se mantiene a la espera de la respuesta de la acción. En función del resultado de esta acción, se actualiza el estado del problema para reflejar los cambios ocasionados por el resultado de la acción realizada.

3.3.3 Descripción general del funcionamiento del sistema

El funcionamiento del sistema se puede dividir en dos fases. La primera, en la que el usuario introduce las metas al sistema y la segunda en la que el sistema resuelve el problema mediante planificación y el robot. A continuación se detalla el funcionamiento del sistema y después se hace énfasis en la etapa en la que el usuario debe introducir las metas interactuando con el robot.

En la Figura 50 se presenta ver el flujo del funcionamiento del sistema que se detalla a continuación. En primer lugar, el usuario inicia el sistema. Una vez que el sistema se haya iniciado, comenzará la fase de almacenamiento de objetos en el sistema (Figura 26), donde en todo momento el robot informará al usuario qué es lo que tiene que hacer. Tras haber almacenado los objetos (metas) en el sistema, el Control iniciará PELEA con la configuración adecuada y esperará hasta que PELEA esté preparado para recibir las metas. Después de recibir la confirmación de PELEA, se le enviarán las metas obtenidas en la fase de almacenamiento de objetos; PELEA introducirá las metas en el sistema de planificación y enviará la primera acción a realizar. En este punto, se repite el siguiente ciclo:

1. El Control espera una acción.
2. PELEA envía la siguiente acción del planificador al Control.
3. El Control recibe la acción, si se trata del la acción final, se da por concluido el problema; en caso contrario, adapta la acción recibida y hace que el robot la ejecute.
4. PELEA se mantiene a la espera hasta que recibe la respuesta de la acción enviada, pudiendo ser: terminada correctamente o no terminada correctamente.
5. Cuando la acción ha terminado, el Control envía a PELEA la respuesta.
6. PELEA recibe la respuesta, compara el estado del problema y obtiene la siguiente acción del planificador.
7. Se vuelve al apartado 1.

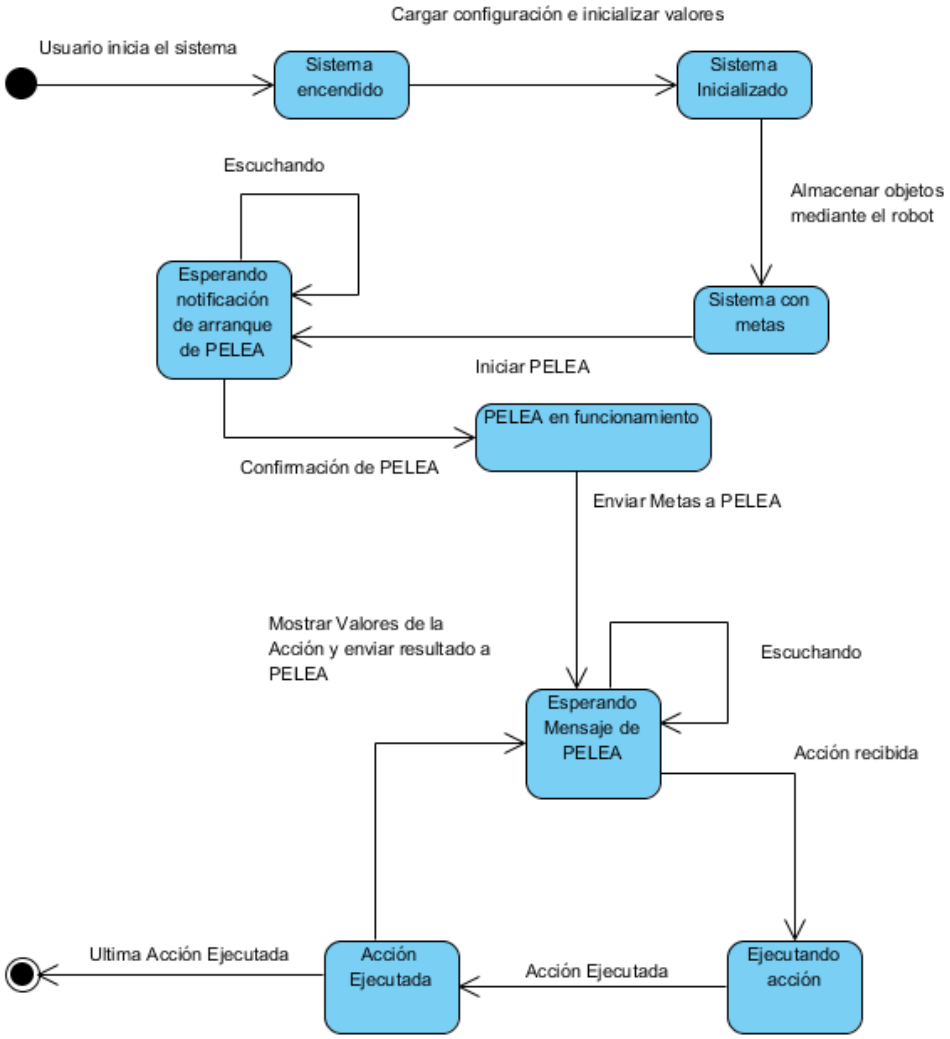


Figura 50 – Funcionamiento del sistema

Capítulo 4: Experimentación

En este capítulo se realiza una descripción de la experimentación llevada a cabo para conseguir el funcionamiento del sistema, también se van a poner los diferentes entornos, pruebas en el simulador y los experimentos que se han realizado para demostrar el correcto funcionamiento del sistema.

4.1 Entorno de pruebas

Las pruebas realizadas durante el desarrollo del proyecto y los experimentos, han tenido lugar en el laboratorio 2.1.B16 y en el pasillo 2.1.B del Edificio Sabatini de la Universidad Carlos III de Madrid, en el Campus de Leganés. Cabe destacar que también se han realizado numerosas pruebas con el simulador Webots For Nao, ya que ofrece un entorno donde realizar pruebas con el robot sin necesidad de trabajar con el físicamente. En la Figura 51 se muestran los elementos utilizados para el entorno de pruebas.

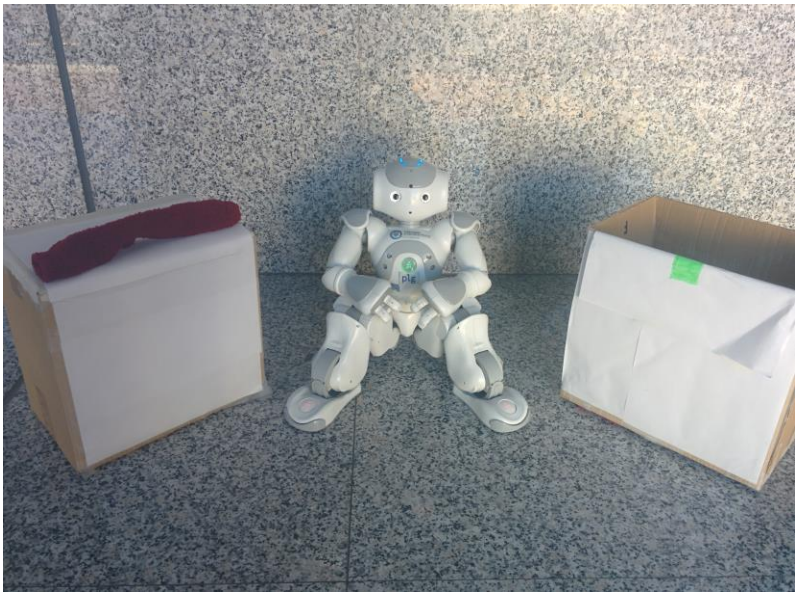


Figura 51 – Entorno de pruebas

Las pruebas de funcionamiento individuales se han realizado únicamente en simulador y en el laboratorio. Mientras que los experimentos se han realizado tanto en el laboratorio como en el pasillo, para poder apreciar como afecta al sistema los diferentes tipos de iluminación y textura del suelo. Ya que se trata de dos entornos diferentes, se van a explicar las diferencias entre uno y otro:

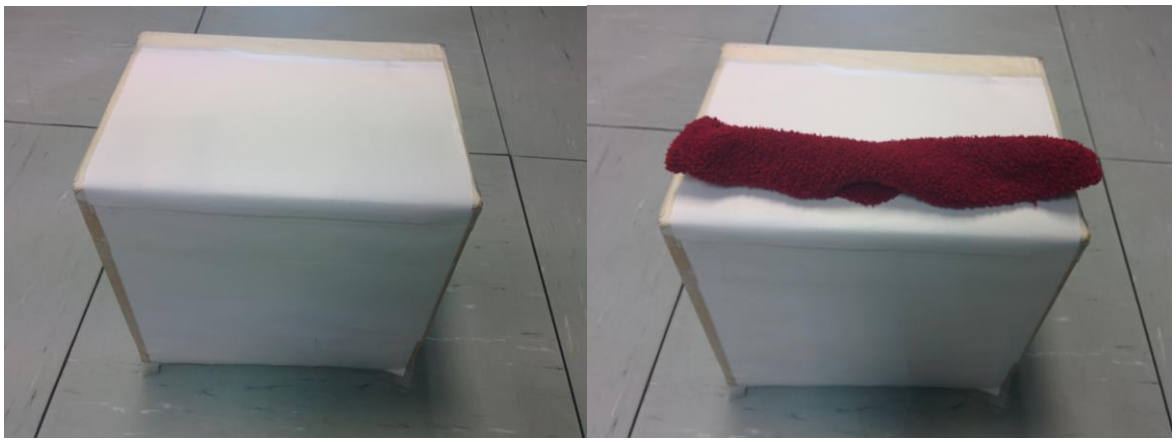
Entorno A (Laboratorio):

- Iluminación artificial mediante lámparas. Se puede ajustar la luminosidad del entorno. Para las pruebas y experimentos la zona siempre ha estado iluminada con al menos dos focos.
- El suelo del laboratorio está compuesto por baldosas algo rugosas con una superficie de plástico, haciendo más difícil que el robot camine ya que para moverse arrastra ligeramente los pies. Adicionalmente, la unión entre estas baldas crea un pequeño desnivel que puede hacer que el robot se tropiece.
- Para este entorno, el tamaño de una cuadrícula se corresponde con un cuadrado de 0.6 x 0.6 metros.

Entorno B (Pasillo):

- Combinación de iluminación artificial mediante lámparas y luz natural. Sólo se puede ajustar la luminosidad del entorno cuando no hay o escasea la luz natural. Para los experimentos la zona siempre ha estado iluminada por luz natural, para reflejar las dificultades que ofrece este tipo de iluminación al sistema.
- El suelo del pasillo está compuesto por baldas de mármol, que hacen que el robot patine ligeramente al moverse.
- Para este entorno, el tamaño de una cuadrícula corresponde con un cuadrado de 0.4 x 0.4 metros.

Para realizar las pruebas y los experimentos se han usado dos cajas iguales(completar con medidas), una como soporte para recoger el objeto y otra para depositarlos. Para que el robot sea capaz de coger el objeto, simplemente habrá que depositarlo extendido cerca de un borde de la caja que tiene unas dimensiones de 30cm largo, 23cm ancho y 31 profundidad. Como se ha mencionado en capítulos anteriores, el objeto utilizado en las pruebas y experimentos se trata de un calcetín. En la Figuras 52(a) y 53(b) se puede ver la caja con y sin calcetín que se ha utilizado en los experimentos.



(a) (b)

FIgura 52 – Caja – soporte del objeto (a) Caja con el objeto colocado (b)

La caja que se ha utilizado como contenedor del objeto ha sido marcada con una pequeña cuadrícula de color verde como se muestra en la Figura 53. De esta manera, se ha podido reutilizar al completo el mecanismo de identificación del objeto.

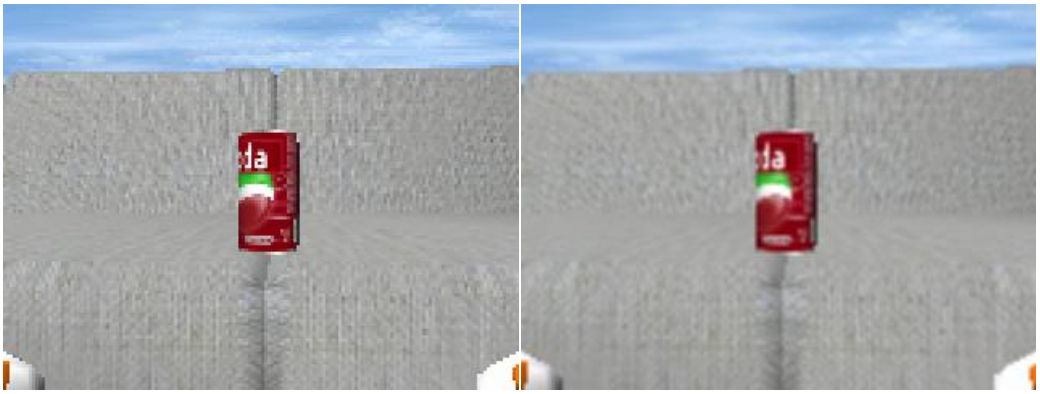
Figura 53 – Caja - contenedor del objeto

4.2 Pruebas del simulador

A continuación se muestran pruebas realizadas en el simulador para comprobar que el reconocimiento de objetos funcionaba adecuadamente. Se pueden apreciar que los pasos son los mismos que los descritos en el apartado 3.3.2.2.4.

(a) (b)

Figura 54 – Imagen original (a) Imagen de OpenCV (b)

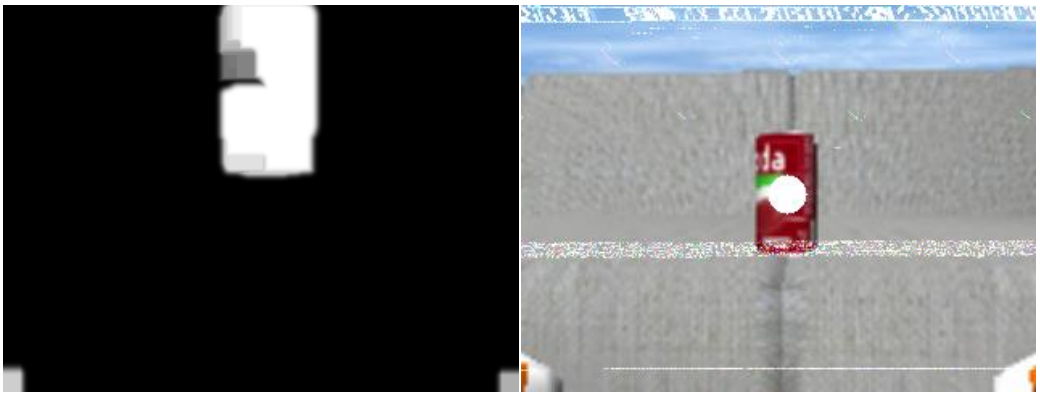


(a) (b)

Figura 55 – Imagen OpenCV cambiada a BGR (a) Imagen difuminada (b)

(a) (b)

Figura 56 – Imagen en formato HSV (a) Imagen en blanco y negro (b)



(a) (b)

Figura 57 – Imagen con suavizado, dilatado y erosionado (a) Centroide encontrado sobre la imagen original (b)

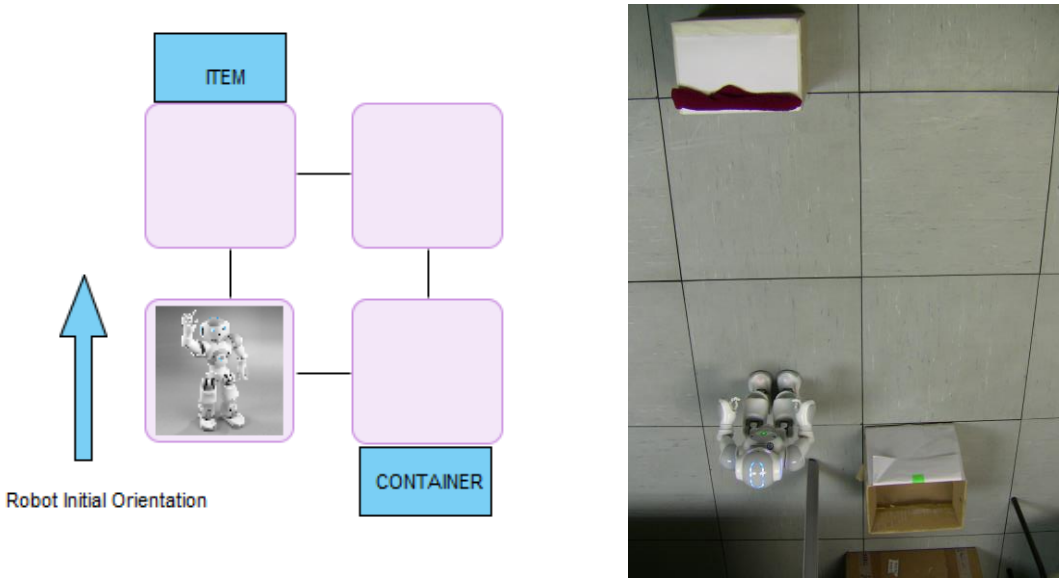
Una vez que esta funcionalidad se implementó y probó en el simulador tal y como se muestran en las Figuras 54-57, se probó con éxito en el mundo real en el entorno A y B de esta sección, sin necesidad de hacer ningún otro cambio en el código. Se puede concluir que el simulador ha sido un entorno adecuado para realizar el desarrollo y testeo del componente de visión, ya que ha funcionado adecuadamente en los entornos A y B sin necesidad de haber realizado el desarrollo con pruebas del mundo real.

4.3 Experimentos

A continuación se presentan los experimentos realizados para poner a prueba el sistema junto con los resultados obtenidos. En las tablas se mostrarán los valores recopilados en los experimentos. En la fila de Resultado, se determinará que ha sido capaz de resolver el problema con un OK, en caso negativo con KO. Para las ejecuciones en las que el sistema no haya sido capaz de resolver el problema (KO), se indicará la razón del fallo mediante un número, ej. KO(1). Debido a la superficie del suelo del entorno B, el robot NAO al moverse se desliza ligeramente, para los experimentos en este entorno se aplicaron unos valores diferentes de corrección, para contrarrestar el deslizamiento.

4.2.1 Experimento 1

El problema a resolver consta de un mapa 2x2 y un objeto. En la imágenes siguiente se puede apreciar la composición del problema. Se trata de un problema sencillo para demostrar que puede resolver un entorno de 2x2.



(a) (b)

Figura 58 – Esquema del problema (a) Imagen real del problema (b)

Razones de fallo:

1. La última medida del giroscopio fue incorrecta por 10º, causando que el robot no se reposicionase y se salió del recorrido.
2. El valor de error de los sensores no superó el mínimo y no se reorientó. La medida del error también contenía fallo.
3. El valor de error de los sensores no superó el mínimo y no se reorientó. La medida del error también contenía fallo.
4. El sistema fallo al detectar el identificador de la caja. El sistema interpretó que estaba 5 centímetros más a la izquierda, haciendo que el robot se desviase. Finalmente el robot dejo el calcetín al lado de la caja en vez de dentro.

Entorno / Valor	A	A	A	A	B	B	B	B
Tiempo de ejecución (segundos)	323.456	392.237	264.510	422.195	269.860	456.360	335.614	312.620
Acciones	8	8	7	8	7	8	8	8
Fotos realizadas	29	39	23	51	19	17	23	18
Número total de reposicionamientos	2	28	17	34	20	18	21	16
Número de reposicinamientos por visión	11	17	8	24	11	8	12	6
Número de reposicionamientos por sensor	1	2	0	1	0	1	0	1
Total de medidas de sensores	2	2	2	2	2	2	2	2
Error medido por sensores (en grados)	16.29	3.05	4.23	23.86	3.07	12.82	3.90	15.19
Resultado	**KO(1)**	**OK**	**KO(2)**	**OK**	**KO(3)**	**OK**	**OK**	**KO(4)**

Tabla 2 – Resultados experimento 1

Para este experimento se puede determinar que este problema es capaz de resolverlo con soltura, ya que para ambos entornos tiene un éxito del 50%. Suelen coincidir los experimentos que más error registran de giroscopio son los que mejor resultado dan o los que más consiguen avanzar en el problema. En cuanto al apartado de visión, ambos entornos ofrecen resultados similares; en el entorno B es normal que tenga menos medidas ya que las aproximaciones las realiza dando pasos más grandes para evitar el problema del deslizamiento producido por el suelo de este entorno. De las razones de fallo, la mayoría se deben a la odometría, ya que no registra el error adecuado en cada giro; sólo una de las razones de error corresponde con visión.

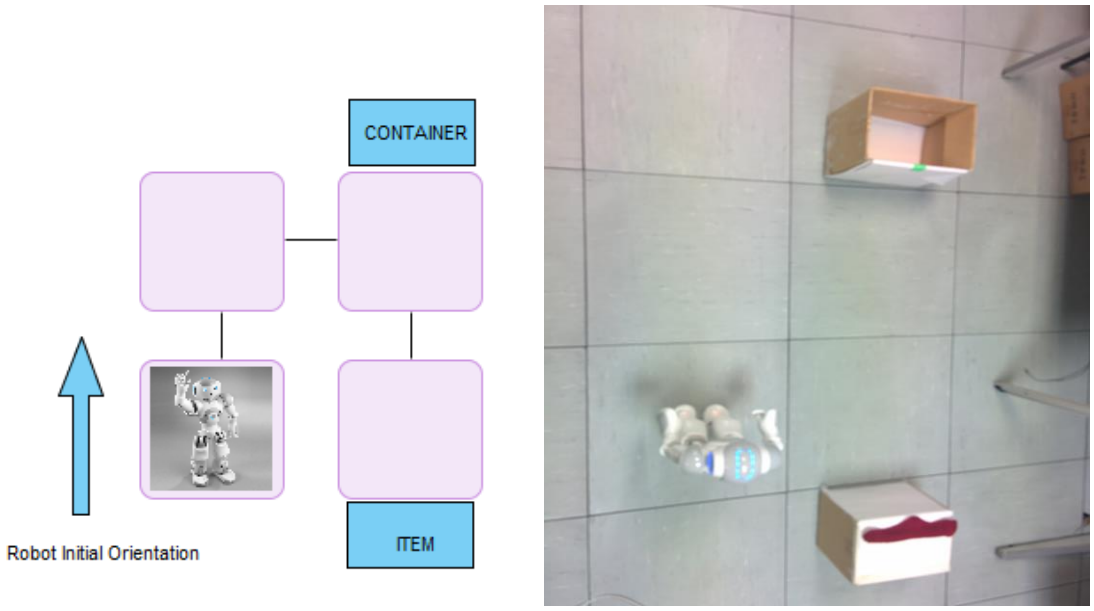
A continuación se adjuntan los enlaces en los que se muestra como el sistema es capaz de resolver el problema en ambos entornos.

Entorno A: https://www.youtube.com/watch?v=5nUpfET_hCE

Entorno B: https://www.youtube.com/watch?v=R9FBfAkmCKY

4.2.1 Experimento 2

El problema a resolver consta de un mapa 2x2 y un objeto. En la imágenes siguiente se puede apreciar la composición del problema. Se trata de un problema algo más complejo para poner a prueba la odometría del sistema, ya que para el tamaño del mapa, tiene una cantidad elevada de giros.



(a) (b)

Figura 59 – Esquema del problema 2 (a) Imagen real del problema 2 (b)

Razones de fallo:

1. El error medido no superó el límite, el robot no se reposicionó, se salió del recorrido.
2. La medida del error fue incorrecta, y giró en el sentido opuesto, se salió del recorrido.
3. Falló al recoger el objeto, al intentar cogerlo lo tiró al suelo.
4. El robot se salió del recorrido porque no reposicionó el error.
5. El robot falló al depositar el objeto porque estaba mal orientado y no vio el identificador de la caja.
6. El robot falló al depositar el objeto por poca distancia ya que no consiguió encontrar el identificador de la caja (estaba desorientado).

Entorno / Valor	A	A	A	A	B	B	B	B
Tiempo de ejecución (segundos)	102.919	099.055	272.505	347.207	102.893	484.126	405.992	306.296
Acciones	7	4	5	11	5	11	11	11
Fotos realizadas	0	0	22	24	0	21	38	12
Número total de reposicionamientos	10	8	22	27	10	25	39	20
Número de reposicinamientos por visión	0	0	11	11	0	13	25	5
Número de reposicionamientos por sensor	1	2	2	4	1	0	2	3
Total de medidas de sensores	2	2	2	4	2	4	4	4
Error medido por sensores (en grados)	7.19	22.50	30.42	40.07	10.42	4.97	21.56	33.79
Resultado	**KO(1)**	**KO(2)**	**KO(3)**	**OK**	**KO(4)**	**KO(5)**	**OK**	**KO(6)**

Tabla 3 – Resultados experimento 2

En este experimento al sistema le cuesta más resolver el problema ya que tiene que realizar más giros que en el Experimento 1. Se nota en el índice de éxito ya que este problema sólo es capaz de resolverlo el 25% del las veces. En las ejecuciones donde el error total registrado por el giroscopio es superior, o bien consiguió resolver el problema o consiguió avanzar hasta la última acción. También se puede observar que las ejecuciones que mejores resultados han dado han sido aquellas en las que más se ha reposicionado el robot. En cuanto a la diferencia de entornos, en este problema el entorno B presenta unos resultados ligeramente superiores. Evaluando las razones de fallo se puede ver que casi todas (5/6) las razones se deben a que el robot no consiguió mantener bien el rumbo y se desorientó o salió del recorrido y sólo una de ellas se debe a que no consiguió coger el objeto.

A continuación se adjuntan los enlaces en los que se muestra como el sistema es capaz de resolver el problema en ambos entornos.

Entorno A: https://www.youtube.com/watch?v=_O1xkDcd2z0

Entorno B: https://www.youtube.com/watch?v=rQ2-VnIGZ-w

Capítulo 5: Gestión del proyecto

El objetivo de este capítulo es describir cómo ha sido gestionado el proyecto. Indicando el modelo de gestión usado y sus fases. También se mostrará la planificación del mismo con un diagrama de Gantt y el presupuesto total del proyecto.

El proceso del desarrollo de software se basará en el modelo de ciclo de vida en espiral [30]. Este modelo de desarrollo combina la simplicidad y formalidad del modelo de cascada con la flexibilidad de los prototipos, que ofrece un buen acercamiento para proyectos en los que no se conoce desde un inicio las técnicas que se van a utilizar para conseguir los objetivos y por ello requiere de un gran número de prototipos hasta dar con el adecuado.

5.1 Descripción de las fases del proyecto

El modelo en espiral (Figura 60) consta de 4 fases principales que son repetidas por cada iteración. Las fases son las siguientes:

1. Planificación: Durante esta fase, se determinan los objetivos de la iteración actual y se realiza un análisis de requisitos. Cada iteración determinará los requisitos que llevarán a completar la funcionalidad necesaria para el proyecto.
2. Análisis y diseño: Esta fase realiza un análisis de diseño y un estudio de viabilidad. Durante ésta etapa se diseñará un prototipo viable.
3. Desarrollo: Durante esta fase, el prototipo es implementado y probado.
4. Evaluación: En esta fase se evalúa el prototipo generado. El principal objetivo de ésta evaluación es detectar defectos del prototipo para planificar los objetivos de la siguiente iteración.

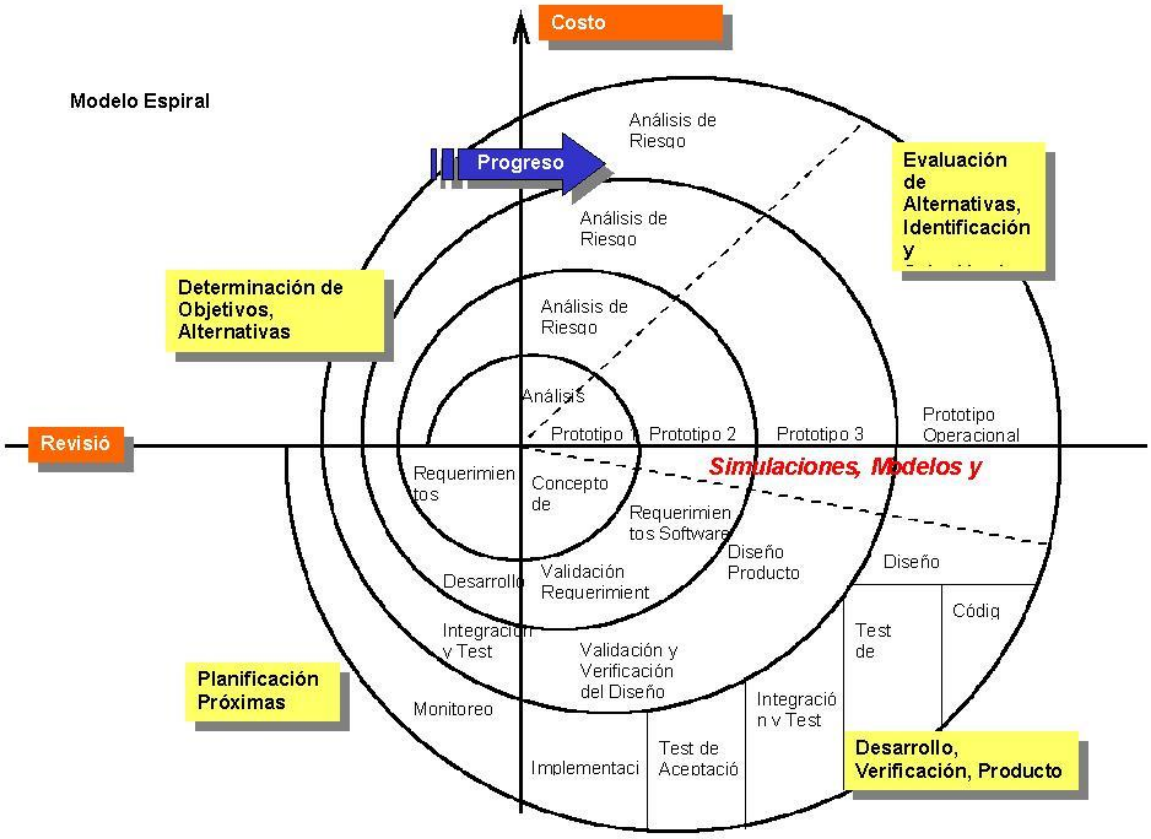


Figura 60 – Modelo de ciclo de vida en espiral

5.2 Planificación

El proyecto se ha realizado en tres iteraciones, cada una de dos meses. En el siguiente diagrama de Gantt (Figura 63) se muestra la evolución del proyecto en todas sus fases. Para esta planificación se ha tenido en cuenta que sólo trabaja una persona 15 horas a la semana (3 horas al día 5 días). La duración total del proyecto es de seis meses y dos semanas (Empezando el 6 de enero y terminando el 14 de Julio).

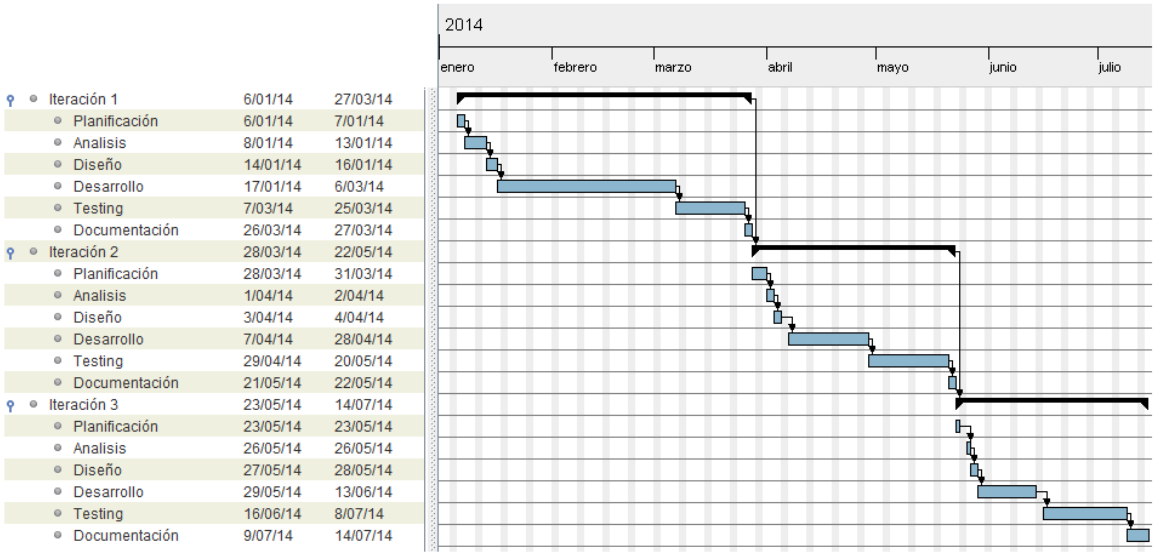


Figura 61 – Planificación de tiempo para el proyecto (Diagrama de Gantt)

5.3 Presupuesto

Los costes del proyecto se detallan en esta sección.

En la tabla 2 se muestra en detalle los costes estimados de los recursos físicos basado en su tiempo de uso (6 meses). Dentro recursos tecnológicos, a los ordenadores portátiles se considera que tienen una vida media de tres años (depreciación del 33% por año), el router se estima que tiene una vida media de tres años y el robot Nao de 5 años (20% depreciación por año). Como el trabajo ha durado 6 meses, el valor se calcula de la siguiente manera: Coste Unitario x Depreciación / 2.

Cantidad	Concepto	Coste Unitario	Total
1	Ordenador Portátil (Macbook Pro)	1200€	200€
1	Robot NAO	12000€	1200€
1	Router Wifi	25€	4,2€
1	Fungibles	50€	50€
			1454,2€

Tabla 4 - Costes estimados de los recursos físicos

En la tabla 3 se muestra en detalle los costes estimas de los recursos humanos basado en la planificación de tiempo de la sección anterior. Los datos de los costes por hora acordes al perfil se han extraído del Boletín Oficial del Estado, del número 22 de enero de 2014, Núm 19 [31].

Perfil	**Costes por hora**	**Horas**	**Total**
Jefe de proyecto	22€	60 horas	1320€
Analista	18€	50 horas	900€
Programador	12,5€	200 horas	2500€
Tester	12,5€	50 horas	625€
Investigador	18€	40 horas	720€
		400 horas	**6065€**

Tabla 5 - Costes estimados de los recursos físicos

Finalmente, la tabla 4 recoge los costes estimados de todo el proyecto.

Concepto	**Total**
Recursos físicos	1454,2€
Recursos humanos	6065€
	7519,2

Tabla 6 - Costes estimados del proyecto

Capítulo 6: Conclusiones y trabajos futuros

En este capítulo se va a hacer un análisis de las conclusiones obtenidas por el trabajo realizado, empezando por unas conclusiones generales, una comparación referente a los objetivos y por último posibles continuaciones al trabajo descrito en los anteriores capítulos.

6.1 Conclusiones generales

Trabajar con sistemas que interactúan con el mundo real es bastante complejo. Aunque estos sistemas estén dotados de sensores para percibir la información del mundo real, esta medición de los sensores lleva error, que se va propagando a lo largo del sistema. Utilizar a un robot bípedo es una labor muy interesante, pero también muy difícil, ya que concretamente el robot NAO, se tambalea mucho al caminar; esto hace que sea muy complicado mantener al robot en una ruta y más aún si tiene que coger objetos o depositarlos. Los sistemas de visión artificial son bastante susceptibles a los cambios de luz y es necesario que se utilicen en escenarios controlados para poder obtener buen resultado de ellos.

Para poder realizar este trabajo ha requerido conocimientos de planificación automática, visión artificial, ingeniería de software, odometría y programación orientada a objetos. El trabajo ha supuesto una mejora personal en las áreas anteriores bien mejorando los conocimientos o creándolos, ya que algunas eran desconocidas para mi.

En este trabajo se han utilizado numerosas técnicas para conseguir el funcionamiento final que se ha descrito, algunas de las tareas probadas y descartadas son las siguientes. Empezando con la recolección de objetos:

- Imagen dividida en 6 áreas: Inicialmente se dividió la imagen en 6 áreas para detectar la ubicación del objeto. Debido a que no era suficientemente preciso, y para facilitar que el robot pudiera coger el objeto con una mano o con otra, se dividió la imagen en 8 áreas, aumentado mucho la precisión.
- También se probaron diversas maneras para coger el objeto. Inicialmente se intentó con el calcetín en posición vertical inspirado por otros proyectos realizados en la universidad Carlos III con este robot. De esta manera era muy complicado posicionar al robot en la manera exacta, ya que si el brazo avanzaba demasiado tiraría el calcetín. Éste método se

descartó y se empezó a colocar el calcetín en forma de bola, pero también se descartó ya que era muy grande como para entrar en la mano del robot. Después se puso en forma de "O", que permitía cogerlo con mayor facilidad ya que se acoplaba a la forma de la mano, pero posicionar al robot era complejo. También se intentó con el calcetín doblado y finalmente con el calcetín estirado, que ha resultado ser la mejor configuración para recogerlo.

- Cuando se acerca mucho un objeto a una de las cámaras del robot, los colores de la imagen capturada por el robot distan mucho de los colores reales, ya que automáticamente realiza un ajuste de brillo. Por este motivo, cuando tiene que reconocer un objeto, no se le puede colocar cerca de las cámaras, al menos hay que guardar una distancia de unos 10 cm. Esto supuso un problema para coger el objeto, ya que el robot necesita una pequeña distancia para poder recoger el objeto, si no, es posible que interprete de manera diferente el color debido al ajuste del brillo.

Para conseguir que el robot fuese capaz de seguir el camino adecuadamente sin salirse, se hicieron numerosas pruebas y se utilizaron las siguientes técnicas:

- Sonar: Se intentó utilizar el sonar para aproximarse a la caja que donde se coloca el objeto. Además el sonar es un sensor con un nivel excesivo de ruido, lo cual impide la realización de tareas que requieren precisión.
- Correción mediante brújula de visón (módulo AlVisionCompass): Utiliza dos imágenes para medir el giro realizado. No se obtenía precisión suficiente como para corregir el error de giro.
- Inicialmente, se corregía el movimiento en línea recta del robot mediante visión siempre y cuando tuviese el objeto delante. Se sustituyó por una corrección fija tras realizar el desplazamiento.

6.2 Conclusiones referentes a los objetivos

Para realizar este apartado se van a presentar los objetivos del la sección 1.3 y se van a presentar las conclusiones para cada objetivo de forma individual.

- Un estudio previo de las técnicas y tecnologías a utilizar para conseguir los apartados expuestos a continuación. Este objetivo se ha logrado, y como conclusión se obtiene que ha sido necesario aprender y formarse en numerosas técnicas y tecnologías, entre ellas:

arquitecturas de control, procesamiento de imagen, representación del color, planificación automática, PDDL, python, OpenCV, API de NaoQi, reconocimiento de objetos y sensado.

- Desarrollo del sistema de movimiento. El robot debe ser capaz de conocer su ubicación (auto-localización) en el entorno de forma aproximada. Así como ser capaz de corregir los errores producidos por sus movimientos de forma automática, mediante la utilización del giroscopio y/o acelerómetro. Este objetivo se ha logrado, como conclusión se obtiene que conocer la ubicación del robot es una tarea compleja ya que se trata de un robot bípedo y es más complicado medir la distancia recorrida que con un robot que por ejemplo, utilice ruedas para desplazarse; además, como normal general el robot no es capaz de caminar en línea recta ya que se desvía ligeramente en su trayectoria, lo cual dificulta más el problema. Adicionalmente, la corrección de los movimientos de forma automática también se trata de una labor realmente difícil, ya que los sensores tienen error en su medición y la obtención de los valores de los sensores no ha sido precisa debido a la velocidad de comunicación entre sistemas (este motivo ha limitado mucho las posibles correcciones mediante sensores); esto ha ocasionado que, con bastante frecuencia, la medición de los valores de los sensores sea incorrecta y el robot corrija mal su error o ni si quiera sea capaz de corregirse. Este fenómeno se puede apreciar en los experimentos realizados, ya que es la causa principal del error de los mismos.
- Desarrollo del sistema de visión. Este debe ser capaz de procesar imágenes para identificar objetos. Este objetivo se ha logrado, como conclusión se obtiene que existen numerosas técnicas para poder identificar objetos, pero la adecuada para este trabajo(por razones de simplicidad, entorno y tipo de objeto a recoger) es mediante el color y gracias a las librerías de OpenCV este proceso se ha agilizado enormemente. Además también he podido apreciar lo sensible que es un sistema de visión artificial a los cambios de luminosidad, distancia y enfoque de la cámara.
- Desarrollo del sistema de recolección de objetos mediante la utilización de los brazos del robot y el sistema de visión del punto anterior. Este objetivo se ha logrado, como conclusión se obtiene que la labor aislada de recolección de objetos mediante la utilización de los brazos del robot es algo complicada debido a las limitaciones de la mano del robot, pero si se le añade que tiene que ser un mecanismo genérico y que además tiene que acercarse al objeto y determinar su posición mediante visión, se convierte en una tarea realmente complicada, que ha requerido de muchas horas de pruebas para

conseguir determinar las distancias y posiciones adecuadas para que el robot fuese capaz de coger un objeto tan simple como un calcetín.

- Integración del sistema de control con la arquitectura PELEA. Este objetivo se ha logrado, como conclusión se obtiene que PELEA ofrece una arquitectura de planificación automática genérica, fácil de adaptar y muy eficaz para el sistema de control de un robot.
- Combinación de todos los sistemas anteriores para crear el sistema de control híbrido. Este objetivo se ha logrado, como conclusión se obtiene que la integración ha requerido de muchas pruebas, para determinar que todos los componentes eran capaces de funcionar de manera conjunta, pero ha sido demostrado el funcionamiento completo del sistema mediante vídeos en los experimentos.
- Evaluación del sistema mediante experimentos para probar su correcto funcionamiento. Este objetivo se ha logrado, como conclusión se obtiene que el sistema funciona adecuadamente en los entornos donde se han realizado los experimentos, pero, que si se aumenta la complejidad del problema, mediante rutas difíciles (muchos giros), incrementar tamaño del problema, o incrementar el número de objetivos, el sistema comienza a fallar de forma incremental.

6.3 Trabajos futuros

Este trabajo se puede extender y mejorar en varios aspectos, algunos de ellos son los siguientes:

- Mejorar odometría: Como se ha podido comprobar por el resultado de los experimentos, la gran mayoría de los fallos se deben a que el robot no gira adecuadamente, y esto lleva a que se salga del circuito. Mejorando la odometría, se podría mejorar en gran medida el rendimiento del sistema. Además permitiría crear más capas de abstracción con el dominio de planificación si se pudiese conocer la distancia exacta recorrida en cada momento.
- Modificar la forma de caminar del robot: Otro de los problemas resaltados en el análisis del sistema, se trata del caminar bípedo del robot. Su forma de caminar hace que se desvíe ligeramente en su trayectoria, incluso con desplazamientos en línea recta, teniendo que aplicar una corrección tras este desplazamiento. Si se mejorase la manera en la que el robot camina, consiguiendo que se mueva en línea recta sin desviación o con muy poca, mejoraría en gran medida el rendimiento del sistema, ya que no habría que aplicar

corrección en los desplazamientos en línea recta y evitaría que el robot se saliera del recorrido.

- Mejorar la visión artificial: Incluyendo reconocimiento de formas, permitiría ubicar mejor la posición del objeto, al menos para conocer sus dimensiones. También sería interesante permitir que el sistema pudiera recoger más objetos.
- Dotar al sistema de sensores de profundidad, mediante la cabeza laser para el robot NAO, ya que mejoraría el mecanismo de aproximación al objeto, haciendo que sea más rápido y que no tuviera que depender tanto de visión. Otra posibilidad sería calcular la profundidad mediante procesamiento de imagen.
- Interfaz hombre-robot por voz: Como indicaba uno de los requisitos (de baja prioridad), se podía haber realizado la interacción entre el usuario y el robot mediante voz, en vez de utilizar los botones táctiles de la cabeza del robot.
- Permitir que el robot pueda coger los objetos con ambas manos y con la mano izquierda: Actualmente el robot sólo cogerá los objetos con la mano izquierda, permitir que los pueda coger también con la mano derecha, o incluso con ambas manos, facilitaría la manera en la que coge los objetos, y en algunas ocasiones, evitaría que el robot fallase al coger el objeto por la posición en la que se encuentre. También, en caso de que el robot fallase al coger el objeto, se podría realizar un intento con la otra mano.
- Habilitar la replanificación: Incorporando en los procesos de definición de estado cierta información del entorno, para poder replanificar o descubrir nuevas metas en el problema.

Capítulo 7: Anexos

7.1 Manual de instalación

El sistema entero está compuesto por un fichero de python .py que puede ser ejecutado directamente por un sistema siempre y cuando tenga instalado los requisitos del entorno operacional de software. Además del ejecutable de python, también se proporciona una carpeta con todas las librerías desarrolladas en este trabajo, necesarias para que el ejecutable de python funcione, un fichero de configuración xml con los parámetros para PELEA, un script para ejecutar pelea, un fichero .jar con la arquitectura PELEA, el planificador Metric-FF, junto con el dominio usado en este trabajo y los problemas utilizados en la experimentación a modo de ejemplo.

7.2 Manual de usuario

Para ejecutar el sistema, hay que editar el fichero de configuración de PELEA configuration4.xml con el fichero de problema que se quiera usar modificando los campos de directorios siguientes.

```
<term value="dir1" name="DOMAIN"/>

<term value="dir2" name="PROBLEM"/>

<term value=”dir3" name="TEMP_DIR"/>
```

Figura 62 – Rutas de los ficheros de PELEA

En “dir” 1 introducir la ruta del dominio de planificación. En “dir2” introducir el problema que se quiera ejecutar y en “dir3” introducir un directorio para crear ficheros temporales del problema y el dominio generados por PELEA. La configuración que se facilita en el fichero no requiere hacer ningún otro cambio, si se quieren realizar cambios adicionales sobre esta configuración, acudir a la documentación de PELEA.

Para ejecutar el sistema bastará con abrir un termnal e ir a la ubicación del archivo donde el fichero .py está ubicado, y ejecutar el siguiente comando: py objectPicker.py –a “IP” –p “port” –c –e “numero”. Donde “IP” se trata de la IP del robot, “port” el puerto y “numero” para cargar un entorno u otro; 0 corresponde con el entorno A de experimentación y 1 corresponde con el B.

7.3 Librerías del NAOqi usadas

Los módulos del NAOqi 1.14.5 [32] que se han utilizado son los siguientes:

- ALMemory: Permite acceder a los valores de memoria del robot.
- ALMotion: Permite controlar el movimiento del robot, incluyendo la rigidez de las articulaciones, caminar, la posición de las articulaciones y sus posiciones en el espacio cartesiano.
- ALNavigation: Permite hacer caminar al robot de manera segura, utilizando el sonar para detectar obstáculos.
- ALRobotPosture: Permite hacer que el robot adopte una postura predefinida.
- ALTextToSpeech: Permite hacer que el robot hable a través de sus altavoces.
- ALVideoDevice: Permite utilizar las cámaras del robot.
- ALVisualCompass: Permite obtener la rotación del robot comparando dos imágenes tomadas con una cámara.
- ALSonar: Permite obtener los valores del sonar del módulo ALMemory.

7.4 Problemas usados en la experimentación

7.4.1 Problema experimento 1:

```
(define (problem obstacles)
(:domain NAO)
(:objects
        north - orientation
        south - orientation
        east - orientation
        west - orientation
        cell00 - cell
        cell01 - cell
        cell02 - cell
        cell10 - cell
        cell11 - cell
        cell12 - cell
        cell20 - cell
        cell21 - cell
        cell22 - cell
        color1 - color
        first - order
        second - order
        third - order

)
```

Figura 63 – Problema 1 experimentación – parte 1

```
(:init
        (linked cell00 cell01 east)
        (linked cell00 cell10 north)
        (linked cell01 cell00 west)
        (linked cell01 cell11 north)
        (linked cell10 cell00 south)
        (linked cell10 cell11 east)
        (linked cell11 cell10 west)
        (linked cell11 cell01 south)
        (robot cell00 north)
        (item color1 cell10 north)
        (container color1 cell01 south)
        (next first second)
        (end-goal second)
        (current-goal first)
        (position color1 first)
)
(:goal
        (and
                (delivered color1)
                (finished)
        )
)


)
```

Figura 64 – Problema 1 experimentación – parte 2

7.4.2 Problema experimento 2:

```
(define (problem obstacles)
(:domain NAO)
(:objects
        north - orientation
        south - orientation
        east - orientation
        west - orientation
        cell00 - cell
        cell01 - cell
        cell02 - cell
        cell10 - cell
        cell11 - cell
        cell12 - cell
        cell20 - cell
        cell21 - cellr
        cell22 - cell
        color1 - color
        first - order
        second - order
        third - order
)
```

Figura 65 – Problema 2 experimentación – parte 1

```
(:init
        ;(linked cell00 cell01 east)
        (linked cell00 cell10 north)
        ;(linked cell01 cell00 west)
        (linked cell01 cell11 north)
        (linked cell10 cell00 south)
        (linked cell10 cell11 east)
        (linked cell11 cell10 west)
        (linked cell11 cell01 south)
        (robot cell00 north)
        (item color1 cell01 south)
        (container color1 cell11 north)
        (next first second)
        (end-goal second)
        (current-goal first)
        (position color1 first)
)

(:goal
        (and
                (delivered color1)
                (finished)
        )
)


)
```

Figura 66 – Problema 2 experimentación – parte 2

Glosario

Actor virtual: Se trata de un elemento que no es un actor, pero por simplificaciones del modelado se trata como un actor.

Application Programming Interface: es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Células fotorreceptoras: Células capaces de percibir la variación lumínica.

Cielab: Espacio de color que se calcula usando raíces cúbicas.

Daemon: Proceso en segundo plano.

Efector final: Parte con la que un robot manipula objetos.

Framework: Conjunto de funciones y clases externas a un sistema sobre las que se puede extender funcionalidad.

Grados de libertad: Ejes en los que puede moverse una estructura o articulación.

Luz emitida: Luz que desprenden los objetos por sí mismos.

Luz reflejada: Luz que desprenden los objetos provenientes de otro foco de luz.

Luz transmitida: Luz que se ha transmitido a través de un medio.

Numpy: Librería de python orientada a calculos numéricos.

OpenCV: Librería de código libre de procesamiento de imágen.

Proxy: Intermediario en las peticiones de recursos.

PyQt4: Librería de Python que proporciona elementos gráficos.

Sistemas embebidos: Sistema de software diseñado para cubrir una o algunas funciones específicas.

Widget: Pequeña aplicación o programa.

Acrónimos

ADL: Action Description Language.

AGV: Automatic Guided Vehicle.

API: Application Programming Interface.

AUV: Autonomous Underwater Vehicle.

BGR: Blue Green Red.

CU: Casos de Uso.

FF: Fast Forward planner.

FRIDA: Fiendly Robot for Industrial Dual-Arm.

GCC: GNU Compiler Collection.

GNU: GNU is Not Unix. (acrónimo recursivo)

HSV: Hue Saturation Value.

IEEE: Institute of Electrical and Electronics Engineers

PDDL: Planning Domain Definition Language

PELEA: Planning, Execution and Learning Architecture

RGB: Red Green Blue.

RQ: Requisitos.

RRM: Robotic Refueling Mission.

SIFT: Scale-Invariant Feature Transform

SURF: Speeded Up Robust Features

sRGB: saturation Red Green Blue.

UAV: Unmanned Aerial Vehicle

YUV: Luminance-Bandwidth-Chrominance

Bibliografía

[1] Ai-Chang, M. and Bresina, J. and Charest, L. and Chase, A and Hsu, J.C.-J. and Jonsson, A and Kanefsky, B. and Morris, P. and Rajan, K. and Yglesias, J. and Chafin, B.G. and Dias, W.C. and Maldague, P.F. "MAPGEN: mixed-initiative planning and scheduling for the Mars Exploration Rover mission," Intelligent Systems, IEEE , vol.19, no.1, pp.8,12, Jan-Feb 2004.

[2] Bear robot rescues wounded troops. British Broadcasting Corporation http://news.bbc.co.uk/2/hi/health/6729745.stm. Accedido el 19/09/2014

[3] Maja J. Mataric. The Basics of Robot Control. http://www-robotics.usc.edu/~maja/robot-control.html. Accedido el 09/09/2014.

[4] John Kelleher. Robot Control Architectures http://www.comp.dit.ie/jkelleher/appliedcomputing/week8/RCA.pdf. Accedido el 19/09/2014.

[5] Rosen, C. A. and Nilsson, N. J. Application Of Intelligent Automata to Reconnaissance, Technical Report . Stanford Research Institute, November 1966. Project 5953 Interim Report 1.

[6] Rodney A. Brooks (March 1986), "A robust layered control system for a mobile robot", IEEE Journal of Robotics and Automation 2 (1): 14–23.

[7] Reinhard Klette (2014). Concise Computer Vision. Springer 1998.

[8] Kelly, A; Nagy, B.; Stager, D.; Unnikrishnan, R., "Field and service applications - An infrastructure-free automated guided vehicle based on computer vision - An Effort to Make an Industrial Robot Vehicle that Can Operate without Supporting Infrastructure," Robotics & Automation Magazine, IEEE , vol.14, no.3, pp.24,34, Sept. 2007.

[9] Xue Zhang; Haibin Duan; Xinge Gao, "Attitude parameters extraction of UAV based on hybrid computer vision and improved artificial bee colony algorithm," Control Conference (CCC), 2013 32nd Chinese , vol., no., pp.3887,3890, 26-28 July 2013.

[10] Nesi, P.; Innocenti, F.; Pezzati, Paolo, "RETIMAC: REal-TIme Motion Analysis Chip," Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on , vol.45, no.3, pp.361,375, Mar 1998.

[11] Jing Chen; Shen, J.-L.; Jian Zhang; Wangsa, K., "A Novel Multimedia Database System for Efficient Image/Video Retrieval Based on Hybrid-Tree Structure," Machine Learning and Cybernetics, 2006 International Conference on , vol., no., pp.4353,4358, 13-16 Aug. 2006

[12] Hongyi Li; Deklerck, R.; Cornelis, J., "Integration of multiple knowledge sources in a system for brain CT-scan interpretation based on the blackboard model," Artificial Intelligence for Applications, 1994., Proceedings of the Tenth Conference on , vol., no., pp.336,343, 1-4 Mar 1994.

[13] Trigueiros, P.; Ribeiro, F.; Reis, L.P., "Generic system for human-computer gesture interaction," Autonomous Robot Systems and Competitions (ICARSC), 2014 IEEE International Conference on , vol., no., pp.175,180, 14-15 May 2014.

[14] Hua Wang; Hudai Fu; Wei Yang, "The Application of Image Recognition into the Automatic Recognition System for Truck Cab Assembly Identifier," Innovative Computing, Information and Control, 2006. ICICIC '06. First International Conference on , vol.2, no., pp.117,120, Aug. 30 2006-Sept. 1 2006.

[15] Hering, Ewald (1872). "Zur Lehre vom Lichtsinne" (Para la doctrina del sentido de la luz). Sitzungsberichte der Mathematisch–Naturwissenschaftliche Classe der Kaiserlichen Akademie der Wissenschaften (K.-K. Hof- und Staatsdruckerei in Commission bei C. Gerold's Sohn). LXVI. Band (III Abtheilung).

[16] George W. Ernst, Allen Newell (1969). GPS: A Case Study in Generality and Problem Solving. Academic Press. 297p.

[17] Peter J. Bentley, Doheon Lee, and Sungwon Jung, editors. Artificial Immune Systems. Springer-Verlag, Berlin Heidelberg, Germany, 2008. 978-3-540-85071-7.

[18] Drew McDermott. The 1998 AI planning systems competition. AI Magazine, 21(2), 2000. The AIPS-98 Planning Competition Committee 1998 and Drew McDermott. PDDL- The Planning Domain Definition Language. Alfonso Gerevine and Derek Long. BNF Desciption of PDDL3.0, October 2005.

[19] Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. Artificial Intelligence, 2(3-4):189–208, 1971. doi:10.1016/0004-3702(71)90010-5.

[20] Edwin P. D. Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR-98), pages 324–332, 1989. 1-55860-032-9.

[21] R. Fikes and N. Nilsson (1971). STRIPS: a new approach to the application of theorem proving to problem solving. Artificial Intelligence, 2:189-208.

[22] fai.cs.uni-saarland.de (2014) Joerg Hoffmann – Homepage of Metric-FF. https://fai.cs.uni-saarland.de/hoffmann/metric-ff.html. Accedido el 10/09/2014.

[23] B. Nebel, The FF Planning System: Fast Plan Generation Through Heuristic Search, in: Journal of Artificial Intelligence Research, Volume 14, 2001, Pages 253 - 302.

[24] Guzmán, C., Alcázar, V., Prior, D., Onaindía, E., Borrajo, D., Fernández-Olivares, J., Quintero, E.: PELEA – A Domain-Independent Architecture for Planning, Execution and Learning. In: 6th Scheduling and Planning Applications Workshop (SPARK 2012), pp. 38–45 (2012)

[25] www.plg.inf.uc3m.es (2014) Grupo PLG Universidad Carlos III de Madrid – Ciclo de ejecución de PELEA. http://www.plg.inf.uc3m.es/pelea/execution.php. Accedido el 10/09/2014.

[26] Chestnutt J., Lau M., Cheung G., Kuffner J., Hodgins J., Kanade, T. "Footstep Planning for the Honda ASIMO Humanoid," Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on , vol., no., pp.629,634, 18-22 April 2005.

[27] Han JingGuang, Campbell Nick, Jokinen Kristiina, Wilcock Graham. "Investigating the use of Non-verbal Cues in Human-Robot Interaction with a Nao robot," Cognitive Infocommunications (CogInfoCom), 2012 IEEE 3rd International Conference on , vol., no., pp.679,683, 2-5 Dec. 2012.

[28] Aldebaran Robotics, “Nao Robot Reference Manual”, Internal Report, 2009.

[29] IEEE (1998). 830-1998 - IEEE Recommended Practice for Software Require- ments Specifications. Institute for Electrical and Electronics Engineers.

[30] Boehm, B. (1986). A Spiral Model of Software Development and Enhance- ment. ACM SIGSOFT Software Engineering Notes, 11:14–24

[31] Boletín Oficial del Estado: miércoles 22 de enero de 2014, Núm. 19. III Otras disposiciones. Ministerio de empleo y seguridad social. Convenios colectivos de trabajo. http://www.boe.es/boe/dias/2014/01/22/pdfs/BOE-A-2014-639.pdf

[32] community.aldebaran-robotics.com (2014) NAOqi 1.14.5 modules APIs. https://community.aldebaran-robotics.com/doc/1-14/naoqi/index.html. Accedido el 15/09/2014.